DEVELOPMENT OF AN IMPROVED EDGE DETECTION ALGORITHM FOR NOISY COLOURED IMAGES USING PARTICLE SWARM OPTIMIZATION

BY

BASHIR OLANIYI SADIQ

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
FACULTY OF ENGINEERING
AHMADU BELLO UNIVERSITY ZARIA, NIGERIA

JULY, 2015

DEVELOPMENT OF AN IMPROVED EDGE DETECTION ALGORITHM FOR NOISY COLOURED IMAGES USING PARTICLE SWARM OPTIMIZATION

BY

Bashir Olaniyi SADIQ, B.Eng. (ABU) 2011

MSC/ENG/21179/2012-2013

sadiqbashirolaniyi@gmail.com

A THESIS SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES,

AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF A

MASTER OF SCIENCE (MSC) DEGREE IN COMPUTER ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

JULY, 2015

DECLARATION

I Bashir Olaniyi SADIQ declare that the work in this dissertation entitled "**Development of an Improved Edge Detection Algorithm for Noisy Coloured Images Using Particle Swarm Optimization**" has been carried out by me in the Department of Electrical and Computer Engineering, Ahmadu Bello University Zaria. The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this dissertation was previously presented for another degree or diploma at this or any other institution.

Bashir Olaniyi SADIQ _____ _____

Signature Date

CERTIFICATION

This dissertation entitled "**DEVELOPMENT OF AN IMPROVED EDGE DETECTION ALGORITHM FOR NOISY COLOURED IMAGES USING PARTICLE SWARM OPTIMIZATION** "by Bashir Olaniyi SADIQ meets the regulations governing the award of the degree Master of Science (M.Sc.) of the Ahmadu Bello University,and is approved for its contribution to knowledge and literary presentation.

Chairman, Supervisory Committee _____ _____
(Dr. S.M Sani) Signature Date

Member, Supervisory Committee _____ _____
(Dr. S. Garba) Signature Date

Head of Department
(Prof. M.B Mu'azu)_____ _____
Signature Date

Dean, School of Postgraduate Studies
(Prof. A. Z Hassan)_____ _____
Signature Date

DEDICATION

This research work is dedicated to Almighty Allah (S.W.T) for his guidance, kindness, mercy and sympathy.

ACKNOWLEDGEMENT

To Almighty Allah for His grace and favour.

I appreciate my lovely parents Alhaji and Hajia F.O. Sadiq for their support morally, financially and otherwise. I am most grateful to the chairman of my supervisor committee, Dr. S.M Sani who patiently and tirelessly walked me through this work. He is not just my teacher but also a role model to emulate. I also thank my co-supervisor and teacher, Dr. S. Garba for the support and encouragement.

To my mentors, Prof. B.J. Bajoga and Prof. M.B. Mu'azu, I say a big thank you for your patience and time for going through my manuscripts.

I also extend my profound gratitude to my siblings Sadiq Rukkayyah, Sadiq Bashirat and Sadiq Lateefat.I also appreciate my cousinsAlatise Lukman and Abulsalam Yekeen for their understanding and kindness.

I am grateful to all my lecturers and colleagues; Dr. T.A. Sikiru, Dr. Y. Jibril, Dr. A.M. S. Tekanyi, Dr B. Jimoh, Dr. K.A. Abu-Bilal, Dr. O. Oladokun, Engr A.S. Abubakar, Engr Y. Ibrahim, Engr B. Yahaya,Engr E. Obi, Engr. Mu'azu, , Engr Mustapha, Engr M. Abdullahi , Engr. A. I. Abdullahi, Engr. S. Musa, Engr. E. Okafor, Engr. Sae'ed , Salawudeen Ahmed Tijjani, Abdulsalam Olaniyan, Abdulkadir Usman, Zakkariya Sikiru Olayinka, Joseph Stephen Soja, Aliyu Daniel, Ekpo Valentine, Mariam Lami and all those whose names are not captured here.

ABSTRACT

This research work presents an improved edge detection algorithm using particle swarm optimization based on vector order statistics. The proposed algorithm was implemented using MATLAB 2013 script. The algorithm addressed the performance of edge detection in images, with a view to minimizing broken, false and thick edges whilst reducing the presence of noise as well as computational time. A collection scheme based on step and ramp edges was developed for the edge detection algorithm, which explores a larger area in the images in order to reduce false and broken edges. The efficiency of this algorithm was tested on two Berkeley benchmark images in clean and noisy environments with a view to comparing results, both visually and quantitatively, with those obtained using proven edge detection algorithms such as the Sobel, Prewitt, Roberts, Laplacian and Canny edge detection algorithms. The algorithm was also applied to facial and remotely sensed images with a view to testing the algorithm on real life images. The Pratt Figure of Merit (PFOM) was used as a quantitative comparison between the developed algorithm and the proven edge detection algorithms. The benchmark value for the PFOM is between 0-1, which shows efficient detection of edges as the value tends towards 1. The quantitative results obtained using PFOM on the test images in clean environment for the Sobel, Prewitt, Roberts, Laplacian, Canny and the proposed edge detection algorithms are 0.4209, 0.4195, 0.4181, 0.7048, 0.8421 and 0.8480, respectively. This showed that the proposed algorithm detected more edges in clean environment as the value obtained is nearest to 1. The PFOM on the test images in noisy environment for the Sobel, Prewitt, Roberts, Laplacian, Canny and the proposed edge detection algorithms are 0.4191, 0.4191, 0.2807, 0.2811, 0.5606 and 0.8458 respectively. This showed that the proposed algorithm detected more edges in noisy environment as the value obtained is nearest to 1. The proposed algorithm achieved a PeakSignal-to Noise Ratio (PSNR) of 57.7320dB in environment containing $\leq 33\%$ of noise level. This result signifies 3% improvement in detection of edges in noisy environment as compared with the proven traditional edge detection algorithms which achieved an average PSNR of 22-35dB.

TABLE OF CONTENTS

TITLE PAGE
DECLARATION ii
CERTIFICATION iii
DEDICATION iv
ACKNOWLEDGEMENT v
ABSTRACT vi
TABLE OF CONTENTS vii
LIST OF FIGURES ix
LIST OF TABLES xi
LIST OF ABBREVIATIONS xii

CHAPTER ONE: INTRODUCTION

1.1 Background 1
1.2 Statement of Problem 3
1.3 Aim and Objectives 4
1.4 Methodology 4
1.5 Significant Contributions 5
1.6 Dissertation Organization 5

CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction 6
2.2 Review of Fundamental Concepts 6
2.2.1 Image processing 6
2.2.2 Gray-scale image 7
2.2.3 Coloured images 8
2.2.4 Image filtering 10
2.2.5 Image convolution 12
2.2.6 Edge detection 13
2.2.7 Review of existing edge detection algorithms 17
2.2.8 Image noise 24
2.2.9 Particle swarm optimization (PSO) 25
2.2.10 Setting particle swarm optimization parameters 29

2.2.11 Vector order statistics 31
2.2.12 Calculating euclidean distance between two pixels 33
2.2.13 Edge intensity 34
2.2.14 Benchmark test images used 35
2.2.15 Real life images used 36
2.2.16 Pratt figure of merit (PFOM) 40
2.3 REVIEW OF SIMILAR WORKS 41

CHAPTER THREE: MATERIALS AND METHODS

3.1 Introduction 50
3.2 Methodology 50
3.3 Profile Modelling of Edge Intensity 51
3.4 Collection Scheme 51
3.5 The Proposed Edge Detection Algorithm 55
3.5.1 Particle swarm optimization algorithm 55
3.5.2 Algorithm for the vector order statistics 57

CHAPTER FOUR: RESULTS AND DISCUSSIONS

4.1 Introduction 61
4.2 Analysis of the Proposed Edge Detection Algorithm 61
4.1.1 Computation time 65

CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS

5.1 Introduction 74
5.2 Summary 75
5.3 Conclusion 76
5.4 Limitations 75
5.5 Recommendations and Further Work 76
REFERENCES 77
APPENDICES 81

LIST OF FIGURES

Figure 2.1: Relationship between Image Processing and Other Fields 7

Figure 2.2: An Illustration of Gray Scale Image 8

Figure 2.3: Illustration of RGB Colour Space as 3-D Cube 9

Figure 2.4: HSV colour space as 3-D Cone 9

Figure 2.5: A 3x3 Filter Kernel 10

Figure 2.6:A 3x3 Convolution Kernel with an Image 12

Figure 2.7: Gradient Direction of an Image 15

Figure 2.8:4 & 8-Neighbors of C 15

Figure 2.9: Arrangement of Neighborhood Pixels 16

Figure 2.10: 8-Connected and Adjacent Pixel 16

Figure 2.11: A 2x2 Kernel of Robert Edge Detection Algorithm 18

Figure 2.12: A 2x2 Kernel of Sobel Edge Detection Algorithm 19

Figure 2.13: A 3x3 Kernel of Prewitt Edge Detection Algorithm 20

Figure2.14: Three Commonly used Laplacian Convolution Kernel 22

Figure 2.15: General Flow Chart of PSO algorithm 27

Figure 2.16: Edge Detection Pixel Arrangement 31

Figure 2.17: A Representation of Step and Roof Edges 32

Figure 2.18: Test Images Used 33

Figure 2.19: Application of the Edge Detection Algorithm to Face Detection 35

Figure 2.20: Sample of Natural Faces 35

Figure 2.21: Remotely Sensed Image Used 37

Figure 3.1: Block diagram of the Proposed Algorithm 48

Figure 3.2: A Collection of Pixels Based on Step Edges 49

Figure 3.3: A Collection of Pixels Based on Roof Edges 50

Figure 3.4: Integer Notation for 8-Neighborhood Pixel in a 3x3 Window 50

Figure 3.5: Collection Scheme for Step Edge 50

Figure 3.6: Collection Scheme for Roof Edge 51

Figure 3.7: A 3x3 Window Used to Find the Gradient of the Image 51

Figure 3.8: Flow Chart of PSO based on VOS algorithm 53

Figure 3.9: Extracted Input Image from Computer Databases 54

Figure 3.10: Output of the Vector Order Statistics 58

Figure 3.11: Output Result of the Suppressed Edges 59

Figure4.1: Output Result on Test Image (1) 61

Figure4.2: Output Result on Test Image (2) 62

Figure4.3: PSNR for Various Noise Levels 63

Figure4.4: Output Result of Applying the Proposed Algorithm on Facial Image 65

Figure4.5: Output Result of Applying Proposed Algorithm on Remotely Sensed Image 66

Figure4.6: Sobel Edge Detection Algorithm in Clean and Noisy environment 67

Figure4.7: Prewitt Edge Detection Algorithm in Clean and Noisy environment 67

Figure4.8: Robert Edge Detection Algorithm in Clean and Noisy environment 67

Figure4.9: Laplacian Edge Detection Algorithm in Clean and Noisy environment 68

Figure4.10: Canny Edge Detection Algorithm in Clean and Noisy environment 68

Figure4.11:Proposed Edge Detection Algorithm in Clean and Noisy environment 68

Figure4.12: Sample Shape Used to Test the Algorithm 69

Figure 4.13:Comparison of Images Produced by Different Algorithms on Sample Shape 70

Figure4.14:Result of the Proposed Algorithm in Visual Comparison with other work 70

Figure 4.15: Result of the Proposed Algorithm in Visual Comparison with other work 70

Figure4.16: Quantitative Comparison Using Pratt Figure of Merit (PFOM) 72

LIST OF TABLES

Table 4.1: Noise Levels with their respective PSNR 63

Table 4.2: Computation Time 64

Table 4.3: Pratt Figure of Merit for Various Edge Detection Algorithm 72

LIST OF ABBREVIATIONS

Acronyms	Definition
PSO	Particle Swarm Optimization
VOS	Vector Order Statistics
PFOM	Pratt Figure of Merit
PSNR	Peak Signal-to-Noise Ratio
SEDA	Sobel Edge Detection Algorithm
REDA	Roberts Edge Detection Algorithm
PEDA	Prewitt Edge Detection Algorithm
LEDA	Laplacian Edge Detection Algorithm
CEDA	Canny Edge Detection Algorithm
DEDA	Proposed Edge Detection Algorithm

CHAPTER ONE

INTRODUCTION

1.1 Background

Edge detection can be defined as the process of identifying set of connected pixels that forms a boundary between two disjoint regions (Gang *et al.*, 2008). It can also be defined as the process of locating and identifying sharp discontinuities in images (Rashmi *et al.*, 2013). It is mostly used in image analysis to preserve image features and partition images into regions of interest. The discontinuities in these images can be caused by (Ghasemi *et al.*, 2011):

i. Discontinuity in depth and/or surface colour and texture.

ii. Reflection of light, Shadows and Illumination.

Edge detection is an image segmentation technique in which images are partitioned into meaningful regions of interest. Some of the practical applications of edge detection algorithms are in face and finger print recognition, location of objects in satellite images, medical images, and computer aided surgery or diagnosis amongst others (Rashmi *et al.*, 2013). One of the most important challenges of edge detection algorithm is to detect the edges in noisy images. Many traditional edge detection algorithms have been developed to overcome noise such as Sobel, Prewitt, Roberts and Gradient based edge detection algorithms etc. (Rashmi *et al.*, 2013). These traditional edge detection algorithms are very fast but they cannot perform well on noisy images. Hence, the significant problem of these edge detection algorithms are displacement, removed edges, false and broken edges(Maini & Aggarwal, 2011). Noise phenomenon is an obstacle in detection of continuous edges as it causes some variation of pixel intensities, thus reducing the performance of an edge detection algorithm in noisy images (Setayesh *et al.*, 2013). It also leads to unclear and displaced edges (Chaudhary & Gulati, 2013). Many edge detection

algorithms have been developed in the literature over the past years to improve precision of recognized edges. However, they still suffer from producing broken edges and false edges due to noise effect (Maini & Aggarwal, 2011). Therefore, an improved edge detection algorithm is required to detect edges with greater continuity in noisy images in order to reduce the shortcomings of traditional edge detection algorithms. In the field of image processing, there exists basically two types of images which are the gray scale and the coloured images. Numerous researchers have developed edge detection algorithms for gray scale images in the past. But in recent times, with improvement in computer capabilities and the increased applications of coloured images there is need to develop an effective edge detection algorithm for coloured images (Haque & Aljahdali, 2013). Some of the applications of the edge detection algorithms are in image segmentation, image compression, face recognition, computer vision, computer surveillance, medical diagnosis, image encryption/communication multimedia and remotely sensed images, amongst others(Vijayarani & Vinupriya, 2013). In the areas of medical diagnosis, satellite images, face recognition, and computer surveillance, representation of images by its edges reduced the amount of data required to be stored whilst retaining useful information in the image.

Most edge detection algorithms process a single pixel on an image at a time and calculate a value which shows the edge magnitude of the pixel, and the edge orientation. Then, a thresholding technique is utilized to recognize if a pixel is an edge or not (Rai & Dutta, 2013) and the result will be a binary image which indicates the location of all existing edges on the original image. The edges detected by these algorithms are not usually linked and there is no relation among the edge pixels. To solve this problem, most edge detection algorithms utilize a linking technique such as the Hough transform, sequential edge linking etc. However, the linking process in such

techniques is not good except for the edges on simple shapes such as circles or lines(Setayesh *et al.,* 2013). Most of the edge detection algorithms weredeveloped to perform on gray level images as compared with coloured images. This is because edge detection in coloured images is a far more challenging task, and the criteria for a good edge detection algorithm are (Maini & Aggarwal, 2011):

i. The optimal detection must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges).
ii. The edges detected must be as close as possible to the true edges.

1.2 Statement of Problem

The traditional edge detection algorithms use limited or small area to detect a pixel as an edge. Edge detection is very crucial in image processing (Ghasemi *et al.*, 2011) and size of the area being considered has strong influence on the accuracy of the detection (Setayesh *et al.*, 2013). The larger the area, the less the sensitivity to noise, but at the same time, the localization accuracy is lower. In order to increase the localization accuracy of the algorithm, there is the need to consider all the edge patterns. However, this increases the computation time and produces broken and false edges. In this research work, a particle swarm optimization edge detection algorithm for coloured images based on vector order statistics is proposed in order to reduce false and broken edges as well as computational time by exploring a larger area in the noisy images

1.3 Aim and Objectives

The aim of this research work is to develop a particle swarm optimization (PSO) edge detection algorithm for noisy coloured images based on vector order statistics with a view to reducing false and broken edges as well as computational time by exploring a larger area in the noisy images.

The objectives are as follows:

i. Development ofacollection scheme for set of pixels in coloured images with a view to reducing false and broken edges in the image.
ii. Development of a particle swarm optimization edge detection algorithm for noisy coloured images based on vector order statistics.
iii. Validation of the proposed algorithm in (ii) and comparison with the traditional edge detection algorithms using Pratt Figure of Merit (PFOM)

1.4 Methodology

The following methodology was adopted in carrying out this research:-

i. Developingacollection scheme to detect edges in coloured images.
ii. Exploring a larger area and examine normally occurring edge patterns in order to increase the localization accuracy of edge detection and determine which edge pixels should be discarded as noise and which should be retained.
iii. Extracting the global structure of edges in order to detect the edges with greater continuity. Hence, determine the exact location of an edge.
iv. Testing the proposed algorithm in MATLAB 2013b image processing toolbox on two coloured images obtained from the Berkeley benchmark image database, remotely sensed image generated using Google earth software and real facial images.
v. Validation and testing of the proposed edge detection algorithm

1.5 SignificantContributions

The existing traditional edge detection algorithms use a single pixel on an image at a time to calculate a value which shows the edge magnitude of the pixel, and the edge orientation. However, this leads to false and broken edges in the generated output edge map. Therefore, the significant contributions of this research work are itemized as follows:-

i Development of a scheme for collection of pixels based on Step and Ramp edges with a view to reducing false and broken edges that exists when generating the output edge map.

ii Application of Vector order Statistics based on collection of pixels for edge detection in noisy coloured images. An improved Pratt Figure of Merit (PFOM) value of 0.09% and 33.7% were obtainedin clean and noisy environments, as compared to the best amongst the existing traditional edge detection algorithms.

iii The proposed edge detection algorithm produced thin and continuous edges in noisy environment and achieved a PSNR of 57.732dB. This represented a 3% improvement in detection of edges in noisy environments as compared with other proven techniques such as the Sobel, Prewitt, Roberts, Laplacian and Canny.

1.6 Dissertation Organization

The general introduction has been presented in chapter one. The rest of the chapters are presented as follows: a detailed review of the fundamental concepts of image processing, edge detection as well as a review of similar research works is presentedin chapter two. Profile modelling of edge intensity, mathematical equations and formulation of the problem are

presented in chapter three. Analysis and discussions of the results are presented in chapter four. Summary, conclusions limitations and recommendations are presented in Chapter Five.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This chapter is divided into two subsections. The first part discussed the fundamental concepts relevant to the study and the second part provided a review of related works.

2.2 Review of Fundamental Concepts

This sub-section presents an overview of concepts fundamental to the research and a review of the existing traditional edge detection algorithms.

2.2.1 Image processing

Image processing can be defined as the analysis and manipulation of a digitized image with a view to getting an enhanced image or to extract some useful information from it(Petrou & Bosdogianni, 1999). Image processing requires an algorithm that accepts an image as an input and produces an image as an output (Solomon & Breckon, 2011). Images are represented as a matrix of square pixels (picture elements) arranged in columns and rows. The need to extract relevant information from images and interpret their contents has been an essential factor in the development of image processing during the past decades. Image processing applications cover a wide range of human activities, such as the satellite application, medical applications, industrial applications, military applications, law enforcement and security, consumer electronics, the

Internet amongst others. (Solomon & Breckon, 2011). An image is simply a two-dimensional or three-dimensional signal digitized as a grid of pixels, the values of which may relate to other properties other than colour or light intensity (Solomon & Breckon, 2011). The information content of pixels of an image can vary considerably, depending on the type of image being processed. In image processing, the most common type of images used are the gray-scale and coloured images. The relationship between image processing and other fields are depicted in Figure 2.1

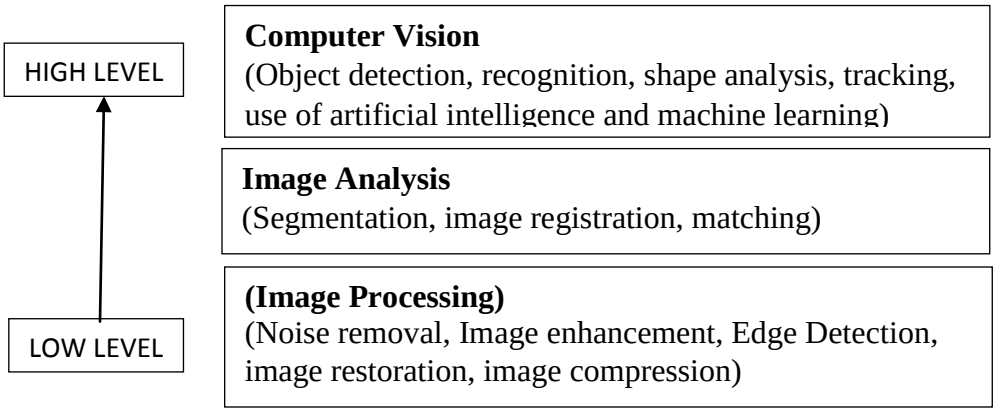


Figure 2.1: Relationship between Image Processing and Other Fields (Petrou & Bosdogianni, 1999)

2.2.2 Gray-scale image

Gray-scale images are 2-D arrays that assign one numerical value to each pixel which is the representative of the intensity at this point (Solomon & Breckon, 2011). Grayscale is a range of shades of gray without apparent colour, consisting of a dark and a white shade. The dark shade represents a black colour which is the absence of transmitted or reflected light, while the light shade represents a white colour which is the total transmission or reflection of light at all visible wavelengths. Intermediate shades of gray are represented by equal brightness levels of the three

primary colours (red, green and blue) for transmitted light. In the case of transmitted light, the brightness levels of the red (R), green (G) and blue (B) (RGB) components are each represented as a number from decimal 0 to 255, or binary 00000000 to 11111111. For all pixels in a red-green-blue grayscale image, R = G = B. The black shade is represented by R = G = B = 0 or R = G = B = 00000000, and the white shade is represented by R = G = B = 255 or R = G = B = 11111111. (Solomon & Breckon, 2011).

An illustration of gray scale image is depicted in Figure 2.2

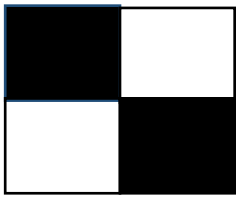


Figure 2.2: An Illustration of Gray Scale Image (Solomon & Breckon, 2011)

2.2.3 Coloured images

Red, Green, Blue, (RGB) or true-colour images are 3-D arrays that assign three numerical values to each pixel, each value corresponding to the red, green and blue (RGB) image channel component respectively (Solomon & Breckon, 2011). True colour images in full spectrum of colours can be represented as a triplet vector, typically the (RGB) components at each pixel location. Here, the colour is represented as a linear combination of the basis colours or values. Considering all the colours that can be represented within the RGB representation, the RGB colour space is essentially a 3-D colour space (cube) with axes R, G. Each axis has the same range 0–255 for the common 1 byte per colour channel (24-bit image representation). The colour black occupies the origin of the cube at position (0:0:0), corresponding to the absence of all three colours and the colour white occupies the opposite corner at the position (1:1:1), indicating the maximum amount of all three colours (Solomon & Breckon, 2011). Colour images are

multivariate in nature, since it assigns three numerical values to each pixel in an image. The ordering of these pixels by arranging the RGB components of the pixels in a vector form is known as the vector order statistics.(Trahanias & Venetsanopoulos, 1996). An illustration of RGB colour space as a 3-D cube is shown in Figure 2.3

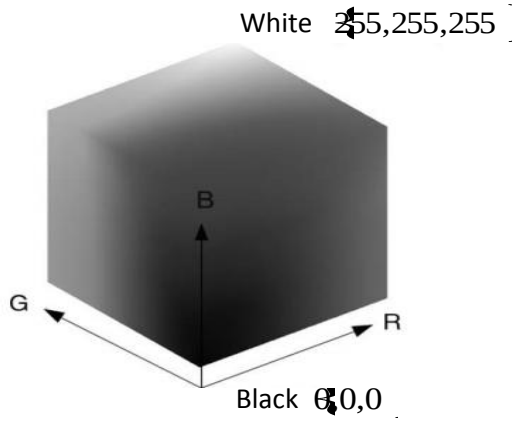


Figure 2.3: An Illustration of RGB Colour Space as a 3-D Cube (Solomon & Breckon, 2011)

The RGB image can also be represented as the hue, saturation and value (HSV) as in Figure 2.4.

Each of these parameters can be interpreted as follows (Chen & Chen, 2010):

i. H (hue) is the dominant wavelength of the colour (red, blue, green)

ii. S (saturation) is the purity of colour

iii. V (value) is the brightness of the colour

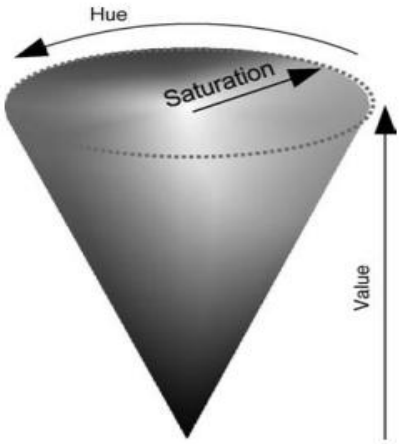


Figure 2.4: HSV Colour Space as a 3-D Cone (Solomon & Breckon, 2011)

2.2.4 Image filtering

Image filters are used to improve the detectability of important image details or objects by man or machine (Petrou & Bosdogianni, 1999). Examples are noise reduction, smoothing, contrast stretching, and enhancement etc. The method of filtering of an image involves the application of a convolution operation to the image in order to achieve the desired output result. The filter is generated by providing a set of weights to apply to the corresponding pixels in a given size neighborhood. The degree of smoothening of an image is controlled using the parameters of the filters (Jahanzeb & Siddiqui, 2013). These set of weights combined produce the filter kernel sometimes called the convolution kernel which is usually represented in a matrix form (Solomon & Breckon, 2011). A typical 3x3 filter kernel is shown in Figure 2.5

$$\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix}$$

Figure 2.5: A 3x3 Filter Kernel (Solomon & Breckon, 2011)

The Gaussian and Kuwahara filters are examples of filters used to remove noise present in images. A Gaussian filter is used to remove Gaussian noise present in an image. A Gaussian noise is a statistical noise having a probability density equal to that of the normal distribution. The Gaussian filter smoothens an image by calculating weighted averages in a filter box. The image is filtered using a discrete kernel derived from a radially symmetric form of the continuous 2-D Gaussian function defined as follows (Solomon & Breckon, 2011):

$$G(p,q) = \frac{1}{2\pi\sigma^2} \exp -(\frac{p^2 + q^2}{2\sigma^2}) \qquad (2.1)$$

Where: σ is the standard deviation of the Gaussian filter

$G(p,q)$ is a 2-dimensional Gaussian function .

The discrete approximations to this continuous function are specified using two free parameters which are the standard deviation of the Gaussian filter σ and the NxN convolution kernel. The degree of smoothing and noise reduction in the image is controlled by the choice of the standard deviation.

The Kuwahara filter is a non-linear smoothing filter used in colourimage processing for adaptive noise reduction. The Kuwahara filter is able to apply smoothing on the image while preserving the edges present in the image. The filter properly removes details even in a high intensity

region, and protects shape boundaries even in low-intensity regions**.** However, the Kuwahara filter is unstable in the presence of noise and suffers from block artifacts (Kyprianidis *et al.*, 2009). The block artifacts in the images are created especially in regions of the image that are highly textured. These blocks disrupt the smoothness of the image and are considered to have a negative effect in the visualization of the image.

2.2.5 Image convolution

Image convolution is simply multiplying the pixels in the neighborhood of each pixels in the image by a set of static weights and then replacing the pixels by the sum of the product. Figure 2.6 shows a 3x3 convolution kernel with an image. The steps in image convolution are summarized as follows (Solomon & Breckon, 2011):

a) Define an appropriate filter kernel to be used
b) Slide the filter kernel over the image so that its center pixel coincides with each pixel in the image.
c) Multiply the pixels lying beneath the kernel by the corresponding values (weights) in the kernel above them and sum the total.
d) Copy the resulting value to the same locations in a new filtered image.

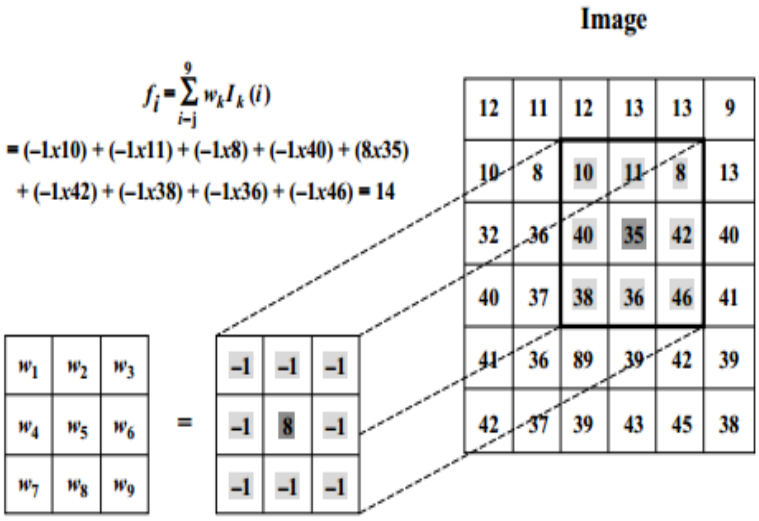


Figure 2.6: A 3x3 Convolution Kernel with an Image(Solomon & Breckon, 2011)

2.2.6 Edge detection

Edges present various definitions in various concepts. An edge can be defined as a single pixel with a local discontinuity in intensity (Saini & Garg, 2012) or can be a boundary which segments an area of an image into two region (Chen & Chen, 2010). Edges are used to extract information about images i.e. location of objects present in the images, their size, image sharpening and enhancement, and mostly image segmentation. Image segmentation is widely used in various fields of image processing, with the aim of partitioning images into either colour based, texture based, depth or motion based and region based with respect to a particular application. It often splits an image into component parts and extracts those parts of interest i.e. objects. Images often includes too much data to be processed, so important features need to be extracted from the image in order to reduce the size of data being processed. Edge detection is a direct way to acquire the outline of the interesting object (Xin *et al.*, 2012).

Terms commonly used in edge detection are highlighted as follows (Gonzalez & Woods, 2008):

i. Pixels

Pixels are the smallest discrete component of an image. It is an abbreviation of the word 'Picture Element' indexed as a column-row (c, r) or (p, q) location from the origin of an image.

ii. **Detection**

This involves the determination of which edge pixels should be discarded as noise and which should be retained. Usually, thresholding provides the criterion used for the detection

iii. **Thresholding**

Thresholding produces a binary image from a gray-scale or coloured image by normalizing pixel values to 1 or 0 depending on whether they are above or below the threshold value(Ashour *et al.*, 2014). A commonly used thresholding technique is the ostu thresholding. The ostu thresholding involves the reduction of gray-level image to a binary image using a binarization algorithm. This is used to separate or segment a region or object within the image based upon its pixel values.

iv. **localization**

This involves the determination of the exact location of an edge. Edge thinning and linking are usually required in this step. Due to the presence of noise in images, the edges detected are thickened. To reduce this problem, an edge thinning technique is implemented known as the Non-Maximum Suppression (NMS).

v. **Non-Maximum Suppression (NMS)**

Non-Maximum Suppression in an image is an edge thinning method used to reduce thick edge responses to thin lines. With the known estimates of image gradients, the Non-Maximum Suppression (NMS) as a task is used to find all the local maxima in an image with respect to the

gradient direction. At every pixel location, it suppresses the edge magnitude of the center pixel by setting its value to 0 if its magnitude is not greater than the magnitude of the two neighbors in the gradient direction. (Pham, 2010).

Edge detection in images often produce thick edges during image processing. The non-maximum suppression (NMS) is simply a process used to reduce the thick edge response in images to produce thin ones during edge detection (Xin *et al.*, 2012). To apply a non-maximum suppression technique to an image in edge detection algorithm, the concept involves splitting the image into four directions. Using an 8 neighboring pixels, 0^0, 45^0, $90^0$and $135^0$are the directions that exists in the image**.** The edge direction is perpendicular if it is running along the 90^0 line. The non-maximum suppression technique also check the left and the right neighboring gradient values. check if the gradient at the present pixel is greater than its left and right neighbors, then it is considered an edge, else it is discarded(Maini & Aggarwal, 2011). This process gives single pixel thin edges.Figure 2.7 shows the gradient direction of an image.

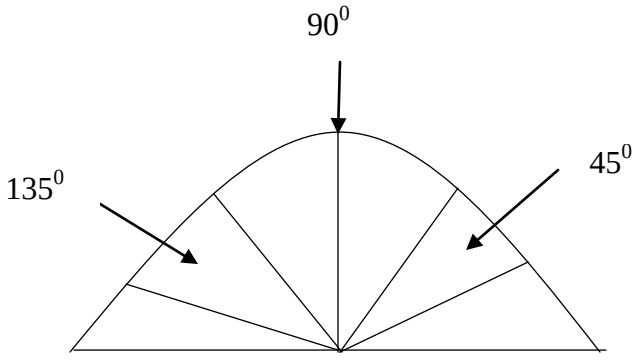


Figure 2.7: Gradient Direction of an Image(Maini & Aggarwal, 2011)

(p-1, q-1)	**(p, q-1)**	**(p+1, q-1)**
(p-1, q)	**(p, q)**	**(p+1, q)**

vi. **Neighborhood pixels**

Any pixel C (p, q) has two vertical and two horizontal neighbors, given by (p+1, q), (p-1, q), (p, q+1), (p, q-1). This set of pixels are called the 4-neighbors of C, and denoted by N_4 (C). Also, the four diagonal neighbors of C (p, q) are given by (p+1, q+1), (p+1, q-1), (p-1, q+1), (p-1, q-1) and denoted by N_D (C). The points N_D (C) and N_4 (C) are together known as 8-neighbors of the point C denoted by N_8 (C) as shown in Figure 2.8. Figure 2.9 shows the arrangement of the neighborhood pixels. Some of the point in the N_4, N_D, and N_8 may fall outside the image if C lies on the border of the image.

(a) N_4 (C) (b) N_8 (C)

Figure 2.8: 4 & 8- Neighbors of C

(p-1, q+1)	**(p, q+1)**	**(p+1, q+1)**

Figure 2.9: Arrangement of Neighborhood Pixels(Solomon & Breckon, 2011)

vii. Connected and adjacent pixels

Two pixels are connected if they are neighbors and their intensity levels satisfy some specified criterion of similarity. For example, in a binary image two pixels are connected if they are 4-neighbors and have same value (0/1). Figure 2.10 shows an example of 8-connected and adjacent pixels.

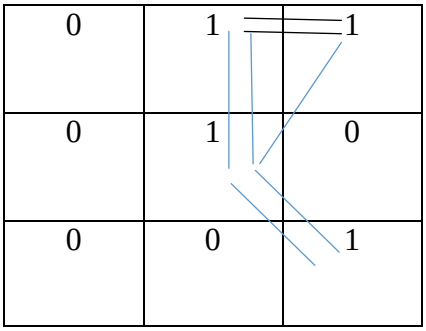


Figure 2.10: 8-Connected and Adjacent Pixels (Solomon & Breckon, 2011)

2.2.7 Review of existing edge detection algorithms

This sub-section describes some of the edge detection algorithms.

i. Gradient-Based Edge Detection Algorithms

The gradient-based edge detection algorithm or first order edge detection algorithm operates by using the first order derivatives to compute the image gradient. For a given image I and the input image I (p, q), the gradient of the image can be calculated using the expression (Rashmi *et al.*, 2013)

$$\nabla I(p,q) = \hat{p}\frac{\partial I(p,q)}{\partial p} + \hat{q}\frac{\partial I(p,q)}{\partial q} \quad (2.2)$$

Where: $\frac{\partial I(p,q)}{\partial p}$ is the gradient of the image in the p direction.

$\frac{\partial I(p,q)}{\partial q}$ is the gradient of the image in the q direction.

Hence, the gradient magnitude of the image can be calculated as(Rashmi *et al.,* 2013):

$$|G| = \sqrt{\left(\frac{\partial I}{\partial p}\right)^2 + \left(\frac{\partial I}{\partial q}\right)^2} = \sqrt{G_p^2 + G_q^2} \quad (2.3)$$

The argument can be computed as(Rashmi *et al.*, 2013)

$$\vartheta = \arctan(G_q / G_p) \quad (2.4)$$

The argument is the gradient direction which is always perpendicular to the direction of the edge. The gradient edge detection algorithm is advantageous because of its simplicity and its ability to estimate edge orientation fast. However, the algorithm is inaccurate with noisy images. This results in improper localization, false and broken edges.

ii. **The Classical Edge Detection Algorithms**

The classical edge detection algorithms are of three types which are the Sobel edge detection algorithm, Robert edge detection algorithm and the Prewitt edge detection algorithm (Rashmi *et al.,* 2013)

a) **Robert edge detection algorithm**

The Robert Edge detection algorithm uses a 2x2 convolution kernel to compute the gradient magnitude and direction. It computes the difference between the sum of squares of the diagonally adjacent pixel using discrete differentiation. Thus finding the approximate gradient in the image. The 2x2 kernel is shown in Figure 2.11

1	0
0	-1

G_p

0	-1
-1	0

G_q

Figure 2.11: A 2x2 Kernel of Robert Edge Detection Algorithm (Rashmi *et al.*, 2013)

Where, G_p andG_q are the convolution kernel in the horizontal and vertical directions respectively. The kernels convolve with the image in the two perpendicular directions, responding maximally to the edges running at 45^0 to the pixel grid. The kernels are hereby applied separately to measure the gradient component in each direction. Due to the small kernel size, the Robert edge detection algorithm is highly sensitive to noise and not compatible with modern technology as applied to colour image processing(Maini & Aggarwal, 2011).

b) Sobel edge detection algorithm

The Sobel edge detection algorithm uses a 3x3 convolution kernels designed to operate in both horizontal and vertical directions denoted as G_p and G_q respectively. One kernel is simply the other rotated by 90^0. The mask used by Sobel edge detection algorithm is shown in Figure 2.12

-1	**0**	**+1**
-2	**0**	**+2**
-1	**0**	**+1**

G_p

+1	**+2**	**+1**
0	**0**	**0**
-1	**-2**	**-1**

G_q

Figure 2.12: A 3x3 Kernel of Sobel Edge Detection Algorithm (Gupta & Mazumdar, 2013)

The kernels can be applied to the image separately, which produce the gradient magnitude of the image in the form (Rashmi *et al.*, 2013)

$$|G| = \sqrt{G_p^2 + G_q^2} \quad (2.5)$$

Where: G_p is the gradient of the image in the horizontal direction

G_q is the gradient of the image in the vertical direction

While the angle of orientation of the edge is computed as

$$\vartheta = \arctan(G_q / G_p) \quad (2.6)$$

Due to the larger kernel size, the Sobel edge detection algorithm is less sensitive to noise as compared to the Robert edge detection algorithm (Maini & Aggarwal, 2011)

c) **Prewitt edge detection algorithm**

The Prewitt edge detection algorithm is very similar to the Sobel edge detection algorithm, but uses a different convolution kernel as depicted in Figure 2.13. The Prewitt edge detection algorithm uses a 3x3 convolution kernels designed to operate in both horizontal and vertical directions denoted as G_p and G_q respectively. It performs better than the Sobel edge detection algorithm (Rashmi *et al.*, 2013). The mask used by Prewitt edge detection algorithm is shown in Figure 2.13

-1	**0**	**+1**
-1	**0**	**+1**
-1	**0**	**+1**

G_p

+1	**+1**	**+1**
0	**0**	**0**
-1	**-1**	**-1**

G_q

Figure 2.13:A 3x3 Kernel of Prewitt Edge Detection Algorithm(Maini & Aggarwal, 2011)

The gradient magnitude of the edge is computed in the form of equation (2.5)

Where: G_p is the gradient of the image in the horizontal direction

G_q is the gradient of the image in the vertical direction

While the angle of orientation of the edge is computed using equation (2.6)

iii. Canny Edge Detection algorithm

The Canny edge detection algorithm is widely used in current image processing with improvements (Rashmi *et al.*, 2013). The Canny edge detection algorithm determines edges in images using optimization process to find the maxima of the gradient magnitude of an image which was earlier smoothed by applying a Gaussian filter. But applying the Gaussian filter can cause image blurring and thus localization of edge detection becomes poor(Setayesh *et al.*, 2013). Given an input image I, the methodology described as follows will be used to obtain the edges from the image using canny edge detection algorithm (Maini & Aggarwal, 2011):

a) Filter out the noise present in the input image using a Gaussian filter before locating and detecting the edges
b) Find the magnitude of an edge by taken the gradient of the image
c) The argument (direction) of the edge is computed using the gradient in both horizontal and vertical directions.

d) From the computation of the edge direction, relate the edge direction to a direction that can be traced in the image.

e) After the edge directions are known, apply Non-Maximum Suppression (NMS) to trace the edges along the edge direction in order to suppress any pixel value (set it equal to zero) that is not considered to be an edge.

f) Use thresholding technique to mark pixel as an edge.

Canny edge detection algorithm is more sensitive to weak edges, producing false edges and unstable boundaries as edges resulting in a corrupted edge map (Akram & Ismail, 2013)

iv. **Second Order Edge Detection algorithm**

The Laplacian edge detection algorithm is a typical example of the second order edge detection algorithm, based on the second order derivatives. The Laplacian convolution kernels are depicted in Figure 2.14. In Laplacian edge detection algorithm, the edges are usually marked at the position where the second order derivative becomes zero. The Laplacian edge detection algorithm L (p, q) of an image with pixel intensity values I (p, q) is denoted by (Solomon & Breckon, 2011):

$$L(p,q) = \frac{\partial^2 I(p,q)}{\partial p^2} + \frac{\partial^2 I(p,q)}{\partial q^2} \quad (2.7)$$

Where: $\frac{\partial^2 I(p,q)}{\partial p^2}$ is the second derivative in the p direction.

$\frac{\partial^2 I(p,q)}{\partial q^2}$ is the second derivative in the q direction

1	**1**	**1**
1	**8**	**1**
1	**1**	**1**

-1	**2**	**-1**
2	**-4**	**2**
-1	**2**	**-1**

0	**1**	**0**
1	**-4**	**1**
0	**1**	**0**

Figure 2.14: Three Commonly Used Laplacian Convolution Kernel (Maini & Aggarwal, 2011)

In order to deal with noisy images, the Laplacian kernel is combined with the Gaussian kernel to produce the Laplacian of Gaussian (LoG) kernel. The Laplacian of Gaussian (LoG) is a gradient based edge detection algorithm that uses the Laplacian to take the second order derivative of the image and identify sharp edges(Jothilakshmi & Rajeswari, 2014). It uses the Gaussian edge detection to reduce noise in the image. However, this method produces responses that do not match edges, thus resulting in 'false edges' and localization error may be severe at curved edges (Rashmi *et al.*, 2013)

The Gaussian function of the edge detection algorithm is given by (Rashmi *et al.*, 2013):

$$G(p,q) = \frac{1}{\sqrt{2\Pi\sigma^2}} \exp -(\frac{p^2+q^2}{2\sigma^2}) \qquad (2.8)$$

Where, σ is the standard deviation of the Gaussian filter.

The Laplacian of Gaussian would be computed using the second derivative as follows (Rashmi *et al.*, 2013):

$$LoG = \frac{\partial^2}{\partial p^2} G(p,q) + \frac{\partial^2}{\partial q^2} G(p,q) = \frac{p^2 + q^2 - 2\sigma^2}{\sigma^4} \exp(-\frac{p^2 + q^2}{2\sigma^2}) \quad (2.9)$$

Where: $\frac{\partial^2}{\partial_p^2} G(p,q)$ is the second derivative of the Gaussian function in the p direction.

$\frac{\partial^2}{\partial_q^2} G(p,q)$ is the second derivative of the Gaussian function in the q direction.

The traditional edge detection algorithms use convolution kernels of NxN matrix, usually ≤ 5 to reduce computational time. Using a small filter kernel size means that only a limited area is considered in the algorithms to mark pixels as edge. The effect of using small area is that the algorithm is sensitive to noise and as such localization accuracy will be lower. The average Peak Signal-to Noise Ratio (PSNR) of the existing traditional edge detection algorithms is between 22-35dB (Rajasekaran & Senthilkumar, 2014) using a median filter for reconstruction. A collection of pixel is therefore required to explore larger area to overcome noise and consider the global structure of edges with a view to reducing false and broken edges in reasonable time.

2.2.8 Image noise

Noise are unwanted, high frequency component, random variation of image intensity which are inherent in digital images. Noise in images occur during either the capturing stage or the transmission stage due to physics-like photon nature of light and thermal energy inside the sensors (Verma & Ali, 2013). The presence of noise in digital images simply means that the pixels in the image shows a different intensity value instead of the original pixels values. In digital images, the number of corrupted pixels will show the quantification of noise present in the

image(Neupane *et al.*, 2012). There exists different types of noise in digital images, depending on the type of disturbance. These types of noise are as follows (Verma & Ali, 2013):

i. The Impulsive noise (Salt & Pepper noise)
ii. Amplifier noise (Gaussian noise)
iii. Multiplicative noise (Speckle noise).

The impulsive noise appear in form of black and white dots in an image. This type of noise occur in the image due to sharp and sudden change in image signal. Dust particles during the image capturing stage or corrupted transmission channel are the major causes of this type of noise (Abdul & Funjan, 2013). A Gaussian noise is a statistical noise having a probability density equal to that of the normal distribution. This type of noise occur during acquisition stage due to the environmental condition (Solomon & Breckon, 2011). The multiplicative noise otherwise known as speckle noise are unwanted signal that worsen the resolution of the active radar and synthetic aperture radar (SAR) images. this type of noise originates due to coherent processing of back scatter signals from different distributed points (Verma & Ali, 2013).

In this research wok, the proposed edge detection algorithm will be tested in noisy environment based on the impulsive noise. This is because this type of noise often occur both at acquisition and transmission stages, due to sharp and sudden changes in image signal relating to the different edge profiles generated for the collection scheme.

2.2.9 Particle swarm optimization (PSO)

Particle swarm optimization algorithm was first introduced in 1995 by Eberhart and Kennedy (Venkata & Babu, 2012). The technique was a population based heuristic optimization problem solving algorithm, which was based on the idea of the social behavior of bird flocking, fish schooling and swarm theory (Roomi & Rajee, 2011).

Particle swarm optimization has five basic parameters which are (Kaur & Singh, 2012):

i Pariticle: Candidate solution to a problem

ii Velocity: Rate of position change

iii Fitness: Best solution achieved

iv P_{BEST}: Best value obtained in previous particle

v G_{BEST}: Best value obtained so far by any particle in the population.

The role of particle swarm optimization is to solve image enhancement problem by tuning the parameters mentioned above with a view to obtaining the best combination according to an objective criterion that describes the contrast of the image. The swarm is initialized randomly, with a group of particles and it then searches for optima by updating through iterations. Two best values are used to update each particle in every iteration. The first one is the best solution of each particle achieved so far known as P_{BEST} and the other is the best solution tracked by any particle among all generations of the swarm known as G_{BEST}(Venkata & Babu, 2012). With respect to the two best values obtained, a particle updates its velocity and position with the help of the following equations (Kaur & Singh, 2012):

$$V_i^{t+1} = W^t.V_i^t + C_1.R_1.(Pbest_i^t - X_i^t) + C_2.R_2.(Gbest_i^t - X_i^t) \quad (2.10)$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \quad (2.11)$$

Where: X_i^t and V_i^t denote the position and velocity of the i^{th} particle at time instance t

W^t is the inertial weight at t^{th} instant of time

C_1 and C_2 are positive acceleration constant

R_1 and R_2 are random values generated in the range [0, 1], sampled from a uniform distribution.

The particle swarm optimization technique is initialized with a view to choosing candidates solution randomly within a search space. The algorithm uses the objective function to determine candidate's solution, thereby operating on the resultant fitness values (Blondin, 2009). The general process of implementing particle swarm optimization algorithm is described as follows (Eberhart & Shi, 2001):

i Initializing the particle by choosing random positions and velocities within the problem space

ii For each of the particle in the problem space, determine the fitness function

iii Comparison of the fitness function with the best value obtained in previous particle (P_{BEST})

iv Comparison of the fitness function with the value obtained so far by any particle in the population (G_{BEST})

v If the current value obtained is better (G_{BEST}), reset G_{BEST} to the current particle array index and value.

vi Change the velocity and position of the particle according to equations (2.10) and (2.11)

vii Repeat from step ii until a good fitness is achieved or maximum number of iteration is reached.

The general flow chart for implementing the particle swarm optimization algorithm is depicted in Figure 2.15

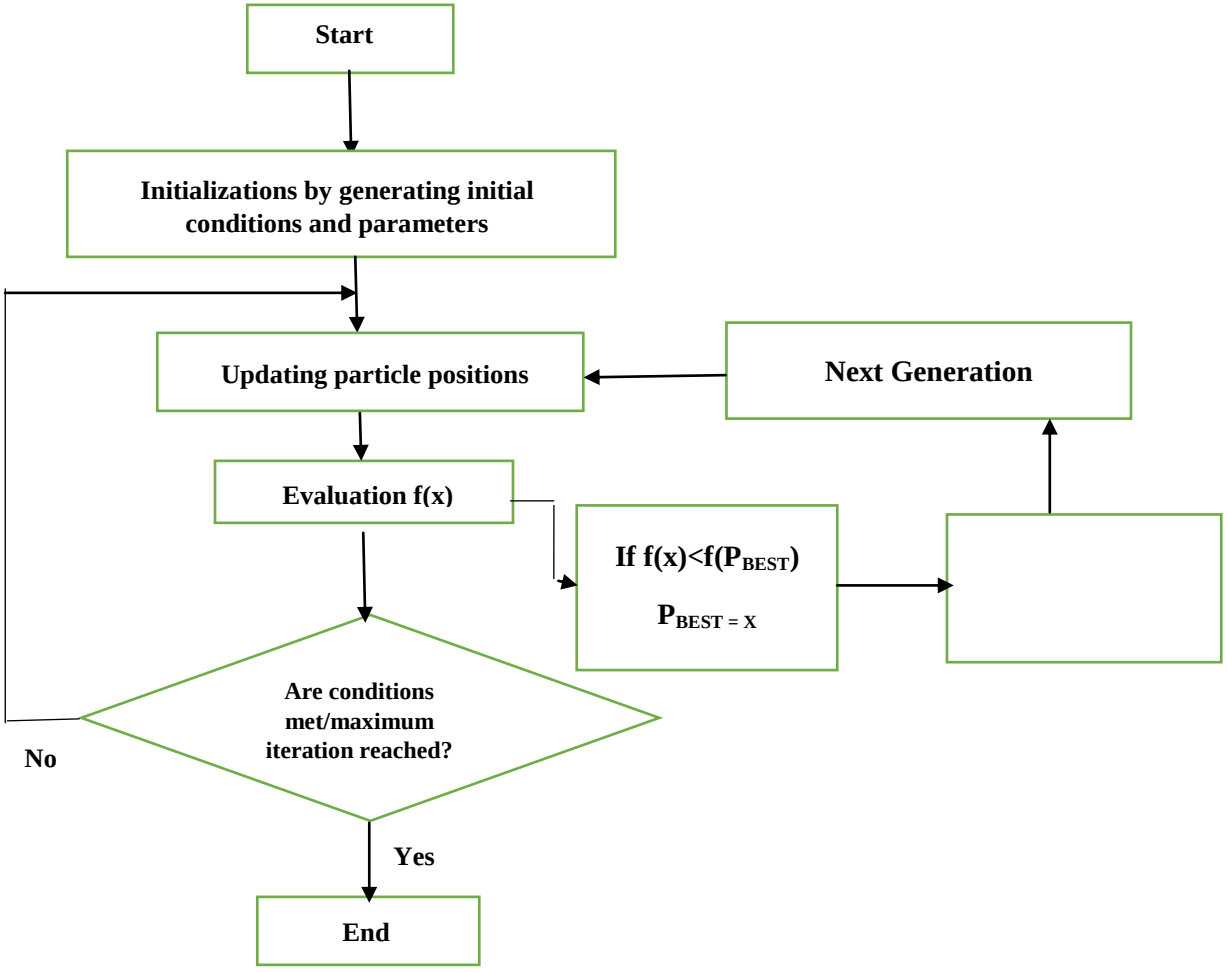


Figure 2.15: General Flow Chart of the Particle Swarm Optimization Algorithm(Eberhart & Shi, 2001)

The inertia weight w and the acceleration coefficient c_1 and c_2 are the particle swarm optimization parameters which are user supplied. The acceleration coefficient are positive

constant within the range of [0 2] while the inertia weight is within the range of [0.8 1.2]. The inertia weight is responsible for keeping the particles moving in the same direction by either suppressing the particles inertia or accelerating the particle in its original ongoing direction(Blondin, 2009).This also controls the particle swarm optimization convergence rate. The values R_1 and R_2 are random values in the range of [0 1]. These values are generated for each velocity update(Shi, 2004).

Some of the advantages of using Particle Swarm Optimization are as follows (Setayesh *et al.*, 2013):

i. Fewer number of parameters: particle swarm optimization is easier to implement because it uses only one parameter which is the velocity calculation.
ii. Using particle swarm optimization algorithm to handle the edge detection in noisy images, does not require any post-processing technique (such as a linking technique).

2.2.10 Setting particle swarm optimization parameters

Digital images are usually affected by noise which is as a result of the sensory devices or the communication media (Roomi & Karuppi, 2010). The particle swarm optimization parameters are set with a view to improving the resolution of the image in the presence of noise by maximizing the information content in the input image. Particle swarm optimization technique is a heuristic approach used to optimize problem based on the social behavior of bird flocking (Venkata & Babu, 2012). The position of the particle is adjusted by a random velocity (V) using equations(2.10) and (2.11)(Zhen-gang & Qin-zi, 2010).

The acceleration coefficient C_1 and C_2 controls how far the particle will move in a single iteration and are usually a random number in the range [0 2]. To improve the performance of the

algorithm, the acceleration coefficient C_1 and C_2 are set to equal integer values. Following (Setayesh *et al.*, 2013) the values w = 0.81, C_1 = 1.4962 and C_2 = 1.4962 are used for the parameters in equation (2.10). The value w is the inertia weight that controls the convergence rate of the particle swarm optimization algorithm. The particle swarm optimization based on vector order statistics edge detection algorithm used a population size of 5 and maximum number of iterations of 10. These parameters are chosen with a view to reducing the computational time during edge detection. In order to measure the quality of a reconstructed image from a noisy environment, a mathematical model is required with a view to determining how much the image has been recovered. These mathematical models are the mean square error (MSE) and peak signal to noise ratio (PSNR) (Ndajah *et al.*, 2011). The mean square error is used as a signal fidelity measure to compare two signals by providing a quantitative score with a view to determining the level of error or distortion between them. The mean square error is calculated using equation (2.12) (Abdul & Funjan, 2013)

$$MSE = \frac{\sum_{p,q}[I_1(p,q) - I_2(p,q)]^2}{p * q} \qquad (2.12)$$

Where; I_1 is the reconstructed image

I_2 is the noisy image

p*q is the dimension of the row to column.

The MSE is however usually expressed as the peak signal to noise ratio measure as in equation (2.13)(Abdul & Funjan, 2013)

$$\text{PSNR} = 10\log_{10}[\frac{T^2}{MSE}] \quad (2.13)$$

Where: T is the range of pixel intensities in an image.

The value of T is calculated using equation (2.14)(Ndajah *et al.*, 2011)

$$T = 2^N - 1 \qquad (2.14)$$

For a unit8 image and unit16 type image of 8-bits and 16-bits respectively, the values of T are computed as:

$T = 2^8 - 1 = 255$ and $T = 2^{16} - 1 = 65535$

Where N is the number of image bits

2.2.11 Vector order statistics

Coloured images comprise of 3 channels which are 8-bits each, denoted by R-channel, G-channel and B-channel. During processing of coloured images, the RGB components in the image is viewed as a vector field that maps a point in the image plane to a three-dimensional (3-D) vector in the colour space. In the three dimensional space, the pixel value is the vector D = (R, G, B), where D is a function of (p, q) in the image.The vector D can be represented at any point using equation (2.15)(Koschan & Abidi, 2005)

$$k = \begin{bmatrix} R_p & R_q \\ G_p & G_p \\ B_p & B_q \end{bmatrix} = (D_p, D_q) \qquad (2.15)$$

The dot product of the vector are represented in the form of a Euclidean vector space as(Wesolkowski & Jernigan, 1999)

$$m \cdot m = U_1^2 + U_2^2 + ...U_n^2 = m^T m = D_{pp}$$

$$n \cdot n = U_1^2 + U_2^2 + ...U_n^2 = n^T n = D_{qq}$$

The distance between two consecutive pixels in the image are calculated using the Euclidean distance (D_E). The Euclidean distance between two image pixel of location m and n is defined as: (Trahanias & Venetsanopoulos, 1996)

$$d(m,n) = \| \text{n} - \text{m} \| \qquad (2.16)$$

Where: ||.|| is the vector norm

The coloured image is viewed as a vector field represented by a discrete valued function. Coloured images are multivariate in nature, and the ordering of a multivariate data as a vector are of four types namely(Trahanias & Venetsanopoulos, 1996); the Vector Range (VR), Minimum Vector Range (MVR), Vector Dispersion (VD) and Mean Vector Dispersion (MVD). Amongst these methods, the Vector Range is easier to implement, efficient but sensitive to impulse noise. In order to deal with noise present in images, the minimum vector was introduced. The minimum vector range is chosen for this research work. The minimum vector range expresses deviation of the vector outlier in the highest rank from the vector median in the window as follows (Koschan & Abidi, 2005)

$$\text{MVR} = \| X_n - X_1 \| \qquad (2.17)$$

Where: X_n is the n^{th} pixel in the image

X_1 is the last pixel in the image

Figure 2.16 shows an example of a 3x3 indexed window.

(1, 1)	(2, 1)	(3, 1)
(1, 2)	(2, 2)	(3, 2)
(1, 3)	(2, 3)	(3, 3)

Figure 2.16: Edge Detection Pixels Arrangement

2.2.12 Calculating euclidean distance between two pixels

Let R, G, B denote the unit vectors along the RGB axis of the RGB colour space. Given an image I, the vector of the colour space in the image can be defined as

$$m = \frac{\partial R}{\partial P} r + \frac{\partial G}{\partial P} g + \frac{\partial B}{\partial P} b \qquad (2.18)$$

$$n = \frac{\partial R}{\partial Q} r + \frac{\partial G}{\partial Q} g + \frac{\partial B}{\partial Q} b \qquad (2.19)$$

(m, n) = size (I) where, size (I) is the dimension of the image used.

The Euclidean distance of a pixel m is given by

$$\|m\| = \sqrt{m \bullet m} \qquad (2.20)$$

Where, m is in the norm form

In a grid of pixels that constitutes the image, the Euclidean distance between two pixels is (m, n) is

$$D_E(m,n) = \|n - m\| \qquad (2.21)$$

Since the input images is a colour image consisting of three components, this can be represented as

$$D_E(m_1, n_2) = \sqrt{(n_{11} - m_{21})^2 + (n_{12} - m_{22})^2 + (n_{13} - m_{23})^2} \qquad (2.22)$$

Where D_E is the Euclidean distance between two pixels

$$D_E(P_{1,1}RGB - P_{1,2}RGB) = \sqrt{(P_{1,1}R - P_{1,2}R)^2 + (P_{1,1}G - P_{1,2}G)^2 + (P_{1,1}B - P_{1,2}B)^2} \quad (2.23)$$

The Vector Range between grids of pixels is calculated by subtracting the last pixel for the first pixel

$$VR_1 = \left|P_{3,2}RGB - P_{1,1}RGB\right|$$

$$VR_2 = \left|P_{2,3}RGB - P_{1,1}RGB\right|$$

$$VR_3 = \left|P_{1,3}RGB - P_{1,1}RGB\right|$$

$$VR_4 = \left|P_{3,1}RGB - P_{1,1}RGB\right|$$

$$VR_5 = \left|P_{2,1}RGB - P_{1,1}RGB\right|$$

$$VR_6 = \left|P_{1,2}RGB - P_{1,1}RGB\right|$$

$$VR_7 = \left|P_{3,3}RGB - P_{1,1}RGB\right|$$

$$VR_8 = \left|P_{2,2}RGB - P_{1,1}RGB\right|$$

$$VR_0 = \left|P_{1,1}RGB - P_{1,1}RGB\right|$$

$$MVR = \min \{VR_0, VR_1, VR_2, VR_3, VR_4, VR_5, VR_6, VR_7, VR_8\} \quad (2.24)$$

Where, MVR = Minimum Vector Range

2.2.13 Edge intensity

Edge detection in images can be represented based on their intensity profile. There are different types of edges in images which are the step, ramp, ridge and roof edges as in Figure 2.17. The types of edges in images are discussed as follows (Sukhendu, 2013):

i. Step edge: The edges present in images are called step if the image intensity changes sharply from one value on one side of the discontinuity to a different value on the opposite side.

ii. Ramp edge: Ramp edges are step edges whose intensity change is not instantaneous but occur over a finite distance.

iii. Ridge edge: The edges present in images are called Ridge if the image intensityvalue changes sharply but return to starting value within some short distance.

iv. Roof edge: The Roof edge are created due to the intersection of surfaces. Roof edges are Ridge edges where the intensity is not instantaneous but occur over a finite distance.

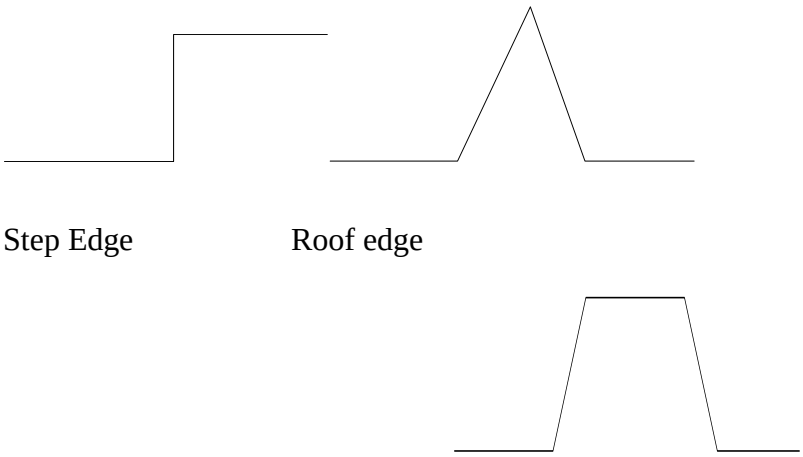


Figure 2.17: A Representation of Step, Roof and Ridge Edges(Sukhendu, 2013)

2.2.14 Benchmark test images used

The test images used in this work are obtained from the Berkeley benchmark image database at www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping. The Berkeley benchmark images are standard test images used in image processing applications to test the efficiency of a developed algorithm by comparison with their ground truth. Ground truth means a set of measurements that is known to be much more accurate than measurements from the system you are testing. The human annotations serve as ground truth for learning as well as a benchmark for comparing different edge detection algorithms. Standard test images are digital image files used

by different algorithm developers to test image processing algorithms with a view to comparing results both visually and quantitatively. These benchmark images are chosen because of the availability of their ground truth and with a view to representing challenges to image processing algorithms such as reproduction of fine details and textures, sharp transitions and edges, uniform regions amongst others. The test images used are depicted in Figure 2.18

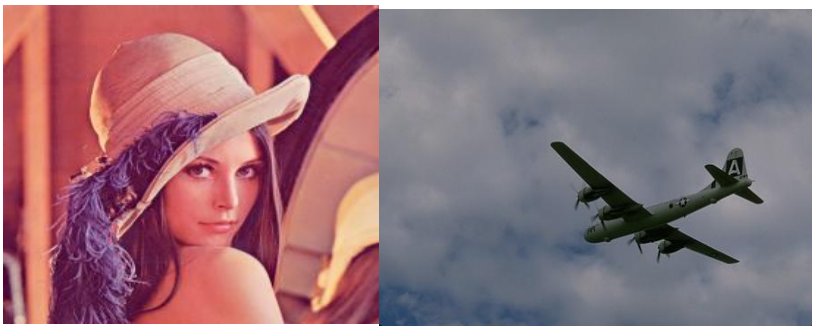


Figure 2.18: Test Images

2.2.15 Real life images used

The edge detection algorithms have a number of applications in real life such as medical diagnosis, face detection, remotely sensed images, and computer surveillance amongst others (Gudipalli & Tirumala, 2013). In this research work, the proposededge detection algorithm using particle swarm optimization based on vector order statistics will be tested on face detection and remotely sensed images due to availability of resources.In applying edge detection algorithm to real life images, the computational complexity increases with the increase in the spatial resolution (Katiyar & Arun, 2014).

i. Face Detection

Face detection is a technique used to find faces at different locations with different sizes in a given location. It is applicable in the field of image processing in biometrics, multimedia

applications, video surveillance amongst others (Anila & Devarajan, 2010). The fundamental steps in face detection and object extraction is successful edge identification and extraction. Edge maps are generated in face detection with a view to representing faces as a single unit. These generated edge maps are one of the most popular ways of representing facial images and their features (Vijayarani & Vinupriya, 2013). The process of face detection are sub divided into three parts namely (Anila & Devarajan, 2010):

i. Extraction of the edge map that constitute the face

ii. Partitioning of the edge map into composite parts for feature extraction

iii. Face verification based on extracted feature

The proposededge detection algorithm is a low level feature that deals with the extraction of pixels that constitutes the faces. The proposed edge detection algorithm helps to facilitate items ii and iii. The block diagram of a simple face detection and extraction system using the proposed edge detection algorithm is shown in Figure2.19

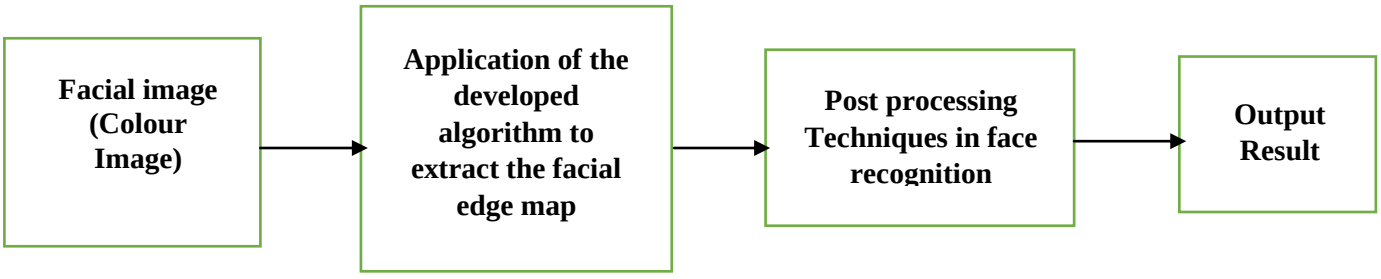


Figure 2.19: Application of Edge Detection Algorithm to Face Detection

The goal of using the proposed edge detectionalgorithm is to determine all important edges related to the facial features such as the pixels that constitute the outline of the ear, mouth, nose and overall face. The proposed algorithm will not identify less significant edge points related to

hair and wrinkles. The proposed edge detection algorithm is applied to three natural faces depicted in Figure 2.20

Facial Image (1) Facial Image (2)

Facial Image (3)

Figure 2.20: Sample of Natural Faces Used

ii Remotely Sensed Images

Remotely sensed images are data that contain important information which are acquired about an object or phenomenon without making physical contact. Remotely sensed images are collections of data on either inaccessible or dangerous areas with a view to monitoring or observing a certain location. Remotely sensed images also replaced expensive and inefficient data collection on ground, assuring that areas of process are not disturbed (Katiyar & Arun, 2014). With the increasing use of remotely sensed imaging applications, the amount of high resolution remotely

sensed image data has a sharp increase. Due to this fact, information generated from these high resolution remotely sensed images cannot be extracted automatically and effectively (Xiong, 2014). The Edge maps generated using edge detection algorithms concentrates on the pertinent information of a remotely sensed image, the method used to extract these edge maps effectively is extremely important for image processing applications (Xiong, 2014). Some of the pertinent information generated by applying the edge detection algorithms on remotely sensed images are road networks, geological features, and desert extraction amongst others (Kaur *et al.*, 2010).

The sample of remotely sensed image used for this research work is obtained using the google earth software due to its availability. The google earth software is a virtual map of the globe that represents geographical information of the earth surface. The google earth software has the ability to display images of different resolution about the earth surface, allowing individuals have a clear view of areas such as cities, roads and other important features without physical presence. Google earth software is installed on the computer system connected to the internet with a view to capturing and representing all locations on the map. Figure 2.21 shows a sample of the remotely sensed image used.

Figure 2.21: Sample of the Remotely Sensed Images Used (Google Earth Software)

2.2.16 Pratt figure of merit (PFOM)

Pratt Figure of Merit (PFOM) provides a quantitative comparison between edge detection algorithms used in image processing. The quantitative measure used by Pratt is determined using equation (2.25)(Akram & Ismail, 2013). The Pratt Figure of Merit analyzes three types of errors which are displaced valid edge points, failed localized edge points and detection of noise flunctuation as edge points.

$$R = \frac{1}{Max(N_I, N_A)} \sum_{k=1}^{N_A} \frac{1}{1 + md^2(k)} \qquad (2.25)$$

Where: N_I is the number of actual edges

N_A is the number of detected edges

m is a scaling constant set to 1/9.

d (k) denotes the distance from the actual edge to the corresponding detected edge

The steps used in the Pratt Figure of Merit are summarized as follows (Pande *et al.*, 2012):

i. Test images are obtained
ii. The edge detection algorithm is applied on the test images
iii. The quality of the resulting edge maps is measured in terms of scalar function

The Pratt Figure of Merit is sensitive to different expected errors, it maximizes when the edge map is perfect and decreases as the error in the edge map increases. The Pratt Figure of Merit measures values between 0 and 1, depending on the quality of the edge detection algorithm used. As the values determined by the Pratt Figure of Merit moves towards 1, it shows best edge detection algorithm (Panetta & Wharton, 2008).

2.3 REVIEW OF SIMILAR WORKS

This sub-section presents a critical and up to date review of similar works undertaken by a number of researchers.

Gang *et al*. (2008) introduced an algorithm for remotely sensed image edge detection based on fuzzy sets. Due to the complexity and noise in remotely sensed images the principle of mathematical morphology edge detection algorithm, sliding window and fuzzy was used to extract the information of edges. The algorithm extracted edges in remotely sensed images and restrained the background noise at the same time by using a sliding window of 8x8 neighborhood surrounding the center pixel. However, the algorithm was applied on a gray level image where the parameters should be set manually for every image. False edges were also frequent in the algorithm.

Dutta & Chaudhuri (2009)presenteda colour edge detection algorithm in RGB colour space. The algorithm in the RGB colour space used a median filter to suppress the noise in the image, then a maximum directional difference of the sum of gray values when each component of the image taken separately were calculated for each pixel. A threshold value is then selected with a view to generating the edge map. In the algorithm,huge set of couple of different colours in the image was transformed into a single value using the RGB transformation procedure. However, in this procedure there are lots of edgesin the colour image that will not be detected due to the transformation technique used. Hence missing edges exists in the generated output edge map.

Gorai & Gosh (2009) presented a gray level image enhancement algorithm using particle swarm optimization technique. Improving the resolution of the image was considered as an optimization problem and particle swarm optimization was used to solve it. The objective criterion for

measuring image enhancement was used which considered entropy and edge information to measure the quality of the image. A transformation function was used to take the intensity value of each pixel from the input image and generate a new intensity value for the corresponding pixel to produce the enhanced image. However, this algorithm was only used to enhance the images i.e. improve the resolution of the image and not detection of edges in images.

Chen & Chen (2010)presented an algorithm for edge detection in RGB colour space. This algorithm used a Kuwahara filter to smoothen the original image before applying edge detection. An adaptive threshold selection method was applied to predict the optimal threshold value. An edge thinning algorithm was used to extract the edges considering each channel independently in RGB colour space. The algorithm used a Sobel edge detection algorithm, which was applied to each channel to generate the edge map using the Otsu's method in selecting the optimal threshold of the main edges and the insignificant edges. The algorithm reduced computational time and could be applied to real-time system where edges are required in a very short time. But with the application of the Kuwahara filter in smoothening process, edges are often displaced or removed due to the presence of noise in the image.

Anila & Devarajan (2010) presented a simple and fast face detection system based on edges. The algorithm used the Sobel filter to extract the edges that constitute the image. The images were divided into blocks with a view to extracting the rectangular features. Facial backgrounds were eliminated with a view to localizing face regions faster. the facial features were fed into a trained Back propagation Neural Network (BPN) to classify the block as either face or

non-face. However with the application of the Sobel filter, the edge map that constitute the image were not extracted completely. Thus, producing broken edges.

Ghasemi *et al*. (2011) created an algorithm to improve edge detection in images, using morphology edge detection algorithm. This edge detection algorithm was considered with various structure elements present in satellite and remotely sensed images. In this algorithm, morphology reconstructor edge detection algorithm was used for processing and removal of noise together with the dilation of the images that are smooth before detecting the edges. Nevertheless, using the structural element in the images shows that the bigger sizes of structure element becomes thicker and broadens. This could cause elongation of the borders and produce thick edges. The algorithm when compared with the traditional edge detection algorithm produced lesser noise and discontinuities, but false and broken edges still existed in the image due to noise.

Xin *et al*. (2012)introduced an algorithm with a view to improving the canny edge detection algorithm to operate on colour images. The algorithm introduced the concept of quaternion weighted average filter (QWAF) and vector analysis to deal with the weakness of the traditional canny edge detection. The algorithm used QWAF with a sliding window of 9x9 to remove the Gaussian noise present in the image, and non-maximum suppression (NMS) based on interpolation for edge thinning. The vector Sobel operator was used to compute the gradient of the image. The performance of the algorithm highly depended on the size of the sliding window. This implies more blurring as well as detecting thicker edges. The outline of broken and false edges appear less using this algorithm but the computation time is increased due to the sliding window.

Venkata & Babu (2012) presented a colour image enhancement using particle swarm optimization technique. This algorithm was an extension of Gorai& Gosh (2009)which was used to enhance gray-level images. The quality of the intensity image was improved using a parameterized transformation function, in which the parameters were optimized by particle swarm optimization based on an objective function. An intensity image from the input colour image was constructed using the hue, saturation and value colour space. The constructed image was taken as the input image and enhancement performed on it. In order to produce the enhanced image, a transformation function containing four parameters namely: a, b, c and k in which each parameters have their defined ranges was used. These parameters were used to find the set of values which produced the best results. The intensity transformation function uses a local and global information of the input image and the objective function considers the entropy and edge information to measure the image quality. However, this algorithm was only used to enhance the images i.e. improve the resolution of the image and not detection of edges in images.

Almadhoun (2013)presented a method for improving and measuring the colour edge detection algorithm in the RGB colour space. The algorithm was an improvement on the work of Dutta & Chaudhuri (2009) with a view to solving the problem of the RGB transformation procedure that resulted in missing edges without increasing the complexity of the algorithm. The algorithm introduced a hue value in the RGB transformation procedure to solve the shortcomings of the Dutta & Chaudhuri (2009) edge detection algorithm. However, the efficiency of the algorithm was tested on only a simple square clean image.The algorithm still produced thick edges.

Rashmi *et al*. (2013)studied and presented a comparative analysis between various existing edge detection algorithms, which were Prewitt, Sobel, Robert and Canny edge detection algorithms. Prewitt, Sobel and Robert edge detection algorithms were classified as traditional edge detection algorithms and are very sensitive to noise but faster and easier to implement. The Canny edge detection algorithm was reported to be better applied to images as compared to the traditional edge detection algorithms due to the fact that it is less sensitive to noise than the traditional edge detection algorithms because of the Gaussian filter used. However, this algorithm still produced false and broken edges due to the presence of noise in the image and highly depended on the value of the adjustable parameter σ which is the standard deviation of the Gaussian filter.

Rai & Dutta (2013)introduced an algorithm for gray-level image edge detection using modified ant colony optimization algorithm. The algorithm generated a pheromone matrix which gave the edge information present at each pixel location of image formed by ants dispatched on image. The algorithm used an improved method based on set of rules which assigned weight to the neighborhood pixel. The pheromone matrix was used in decision making to consider a pixel as edge by using the Otsu's method of thresholding. The algorithm was compared with the Sobel, Roberts, Prewitt and Canny edge detection algorithms. It performed better than the traditional edge detections, due to the fact that it can enhance weak edges and worked well in noisy environment. However, the algorithm was very slow because it required many parameters to be set manually.

Gupta & Mazumdar (2013)presented an extension of the Sobel edge detection algorithm for images. The standard Sobel edge detection algorithm used a 3x3 convolution kernel on an image.

This was extended to a 5x5 convolution kernel in order to deal with noise present in the image. Using a Sobel edge detection algorithm is relatively inexpensive in terms of computation time. However, the gradient approximation it produces is inaccurate, thus false, broken and thick edges exists in the output image.

Haque & Aljahdali (2013)presented an approach for segmenting coloured images using edge detection and thresholding methods. This approach was used to locate objects of interest, separating the pixels that constitutes the edges in the images. Two samples of images were used, converting the first one to gray scale and the other is still used as a colour image. The segmentation method was then applied to the different samples of the images based on edge detection and thresholding methods for comparison. The RGB component in the coloured images are taken independently and edges are detected in each individual component. The individual edges detected in each component are combined to produce the edge map. The edge detection procedure used were the Sobel, Prewitt and Canny algorithms. The algorithms established the fact that it performed better on coloured images as compared to gray scale images.But, the method is inaccurate as it leads to displaced edges. It was also highly dependent on noise present in the images thus producing false edges.

Setayesh *et al*. (2013) proposed a novel algorithm for detecting edges in gray-scale noisy images using particle swarm optimization. The algorithm presented an optimization model for detecting continuous smooth edges using particle swarm optimization to search for good solutions. The result obtained was compared with other existing edge detection algorithm. Although, the algorithm performed better than others in noisy images, it is highly dependent on size of the

square, minimum length of the particle and the weight of the different factors in the fitness function. Consequently, there is need to initialize the particle with more diversity factor which may help the particle swarm optimization convergence. Also the algorithm can only be applied on gray scale images.

Jahanzeb & Siddiqui (2013)presented the analysis of edge detection algorithms for feature extraction in satellite images. The work compared two edge detection algorithms which are the bilateral filter based edge detection algorithm and the canny edge detection algorithm. The research work surveyed the effect of high order Gaussian derivative in edge detection algorithms as applied to satellite images. It was concluded that the bilateral filterbased edge detection algorithm outperform the canny edge detection algorithm and generates a well localized edge map. However, the noise removal capability of the methods was not strong, which makes it highly sensitive to noise in the images. This leads to displaced, false and broken edges in the image.

Vijayarani & Vinupriya (2013)presented a performance analysis between the canny edge detection algorithm and the Sobel edge detection in image mining for face recognition. The research used the canny edge detection algorithm and the Sobel edge detection algorithm to extract edges from facial images with a view to detecting faces. The canny edge detection algorithm outperforms the Sobel edge detection algorithm in face recognition. But the algorithms are sensitive to noise present in the images thus producing false and broken edges.

Chaudhary&Gulati (2013) proposed a new set of filter kernels for segmenting digital images using edge detection. The method used two common traditional edge detection algorithms which are the Laplacian edge detection algorithms and the Sobel edge detection algorithms. The algorithm was tested on grey scale images with a view to comparing the efficiency of the algorithm. But the algorithm produced high false edges in both cases due to blurred and low resolution images.

Gudipalli & Tirumala (2013)presented a comprehensive edge detection algorithm for satellite images. The algorithm was tested on different colour model such as RGB, XYZ and YUV. The RGB component of the images are divided into R, G and B layers of the image. A 3x3 laplacian mask is used to convolve the individual component of the image with a view to detecting the edges in the image. But the algorithm is sensitive to noisy image, failed to produce thin and continuous edges due to the application of the laplacian mask.

Ashour *et al.*(2014)introduced a new method of thresholding for edge detection in digital images. The algorithm used two main phases which are the global and local enhancement phases of the threshold and detection values. A thresholding procedure based on the Shannon entropy was introduced in the algorithm to perform on gray scale images. Though the performance of the algorithm excels as a shape detector. However, using this algorithm cannot process colour images directly unless being converted to gray scale imagesand it failed to produce thin edges.

Jothilakshmi & Rajeswari (2014)presented a modifiedalgorithm for edge detection using the ant colony optimization. The approach used a Laplacian operator to find the edge magnitude, instead of using a pheromone matrix that represents edge information at each pixel location on

the routes formed by ants dispatched on the image. The edges in the images are determined by applying a thresholding scheme on the final pheromone matrix. However, the algorithm required many parameters to be set manually and was performed only on gray scale image.

Katiyar & Arun (2014) presented a comparative analysis of common edge detection techniques in context of object extraction in satellite images. The comparison has been obtained between the existing traditional edge detection algorithms such as the Sobel, Prewitt, Canny and the Laplacian. The investigation showed that the complexity increases with increase in spatial resolution for edge detection methods. It was concluded that the canny edge detection algorithm performs better than other method with lesser false edges. It was also confirmed that the traditional edge detection algorithms are affected by shadows and salt & pepper noise, which needs to be eliminated before detection of edges. However false and broken edges still exists in these techniques.

In view of the imperfections associated with the reviewed works, and the need to overcome the shortcomings of the traditional edge detection algorithms which are false edges, broken edges and thick edges, an improved edge detection algorithm is hereby proposed. The proposed algorithm is expected to reduce noise, false and broken edges. The proposed algorithm used particle swarm optimization based on vector order statistics in order to explore a larger area in the image.

CHAPTERTHREE

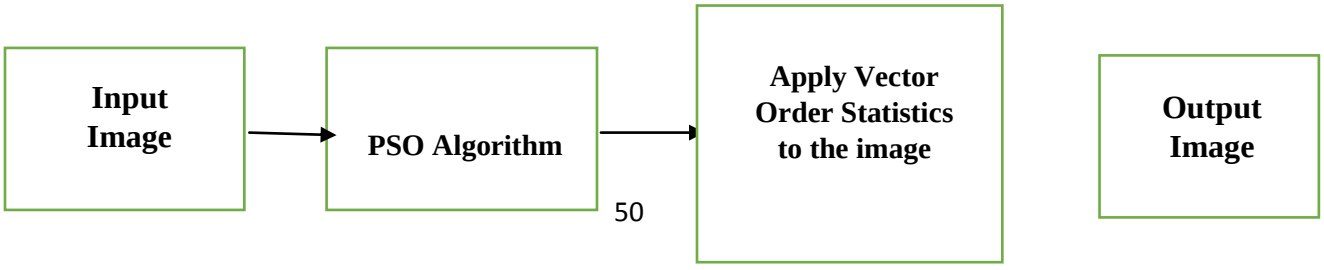
MATERIALS AND METHOD

3.1 Introduction

This chapter presents a detailed profile modelling of edge intensity, mathematical equations and computing parameters for the edge detection algorithm.

3.2 Methodology

The methodology adopted is based on the minimum vector range order statistics (MVR) due to its simplicity and ability to function in noisy environment as applied to coloured images.The Particle Swarm optimization algorithm is used to reconstruct the image from a noisy environment to a clean environment with a view to reducing false edges. Theproposedcollection scheme is then used to find the gradient of the image. The block diagram of the algorithm is shown in Figure3.1



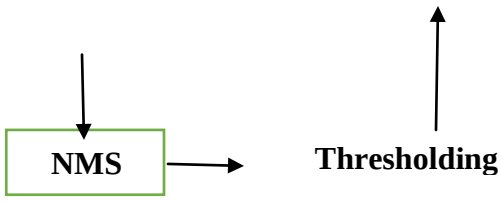


Figure3.1:Block Diagram of the Proposed Algorithm

3.3 Profile Modelling of Edge Intensity

A scheme for the collection of pixels based on step and roof edges is proposed for the algorithm with a view to selecting intensity profiles that changes sharply in images and also with those that change due to the intersection of surfaces. The method processes a collection of pixels at a time instead of a single pixel in order to extract the global structure of the real edge and considers a large area rather than a small one with a view to overcoming false and broken edges. The edges are modelled with respect to the step and roof diagram in Figure 2.17

3.4 Collection Scheme

Due to the collection of a set of pixels in the image, the best edges might lie on a curve. Therefore, the curve passing through the pixels are encoded using the pixel directions as they move from one position to the other. This curve is encoded using the direction of movement from a pixel to the next pixel on the curve. If the number of pixels on a curve is less than the dimension of the particle, the remainder will be set to zero where the dimension of vector representing the particle depends on the image size.In order to detect these edges effectively, a collection of a set of pixels is presented for coloured images in Figures3.2 and 3.3. Figures3.5 and 3.6 show a collection scheme after collection of a set of pixels with respect to the Step and Roof edges.

The collection of pixels based on step and roof edges with Q as the center pixelare as follows.

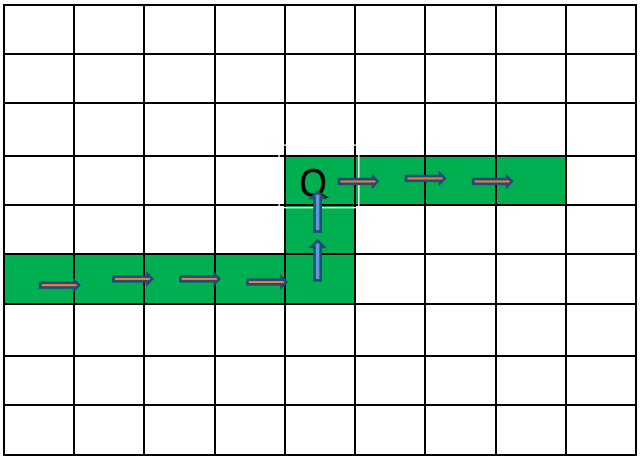


Figure3.2: A Collection of PixelsBased on Step Edges

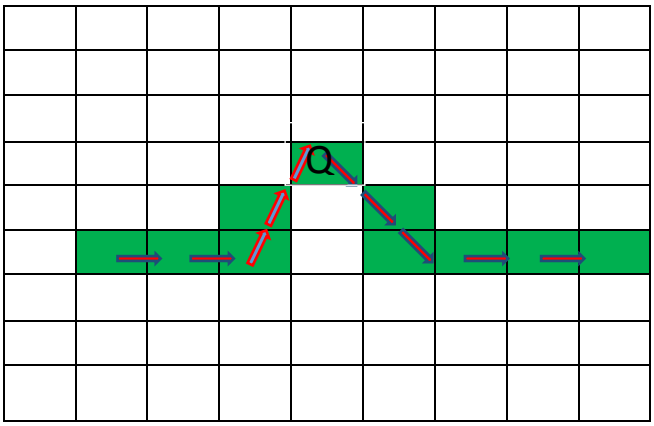


Figure3.3: A Collection of Pixels Based on Roof Edges

The collection of pixels for each collection scheme are from integers 0-8 which implies that the collection of pixels are that of an 8- Neighborhood pixel. This is as shown in Figure3.4

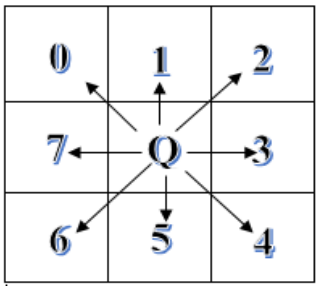


Figure3.4:Integer Notation for 8-Neighborhood Pixel in a 3x3 Window.

Based on the collection of pixels with respect to step and roof edges, the collection schemes in Figure3.5 and Figure3.6 are proposed respectively.

3	**4**	**4**	**4**	**4**	**3**	**4**	**4**	**4**	**0**	**0**	**0**	**…….**

Figure3.5:Collection Scheme for Step Edge

3	**4**	**4**	**3**	**4**	**5**	**4**	**4**	**4**	**0**	**0**	**0**	**……..**

Figure3.6: Collection Scheme for Roof Edge

The proposed collection scheme is represented as a 3x3 window and used to find the gradient in the image. The collection scheme derived from the roof edge is used to compute the gradient of the image along the vertical axis and the collection scheme derived from the step edge computes the gradient of the image along the horizontal axis. This is as shown in Figure3.7

3	**4**	**4**
3	**4**	**5**
4	**4**	**4**

F_x

3	**4**	**4**
4	**4**	**3**
4	**4**	**4**

F_y

Figure3.7: A 3x3 Window Used to Find the Gradient of the Image.

The proposed algorithm in this research is described with the aid of the flow chart shown in Figure3.8

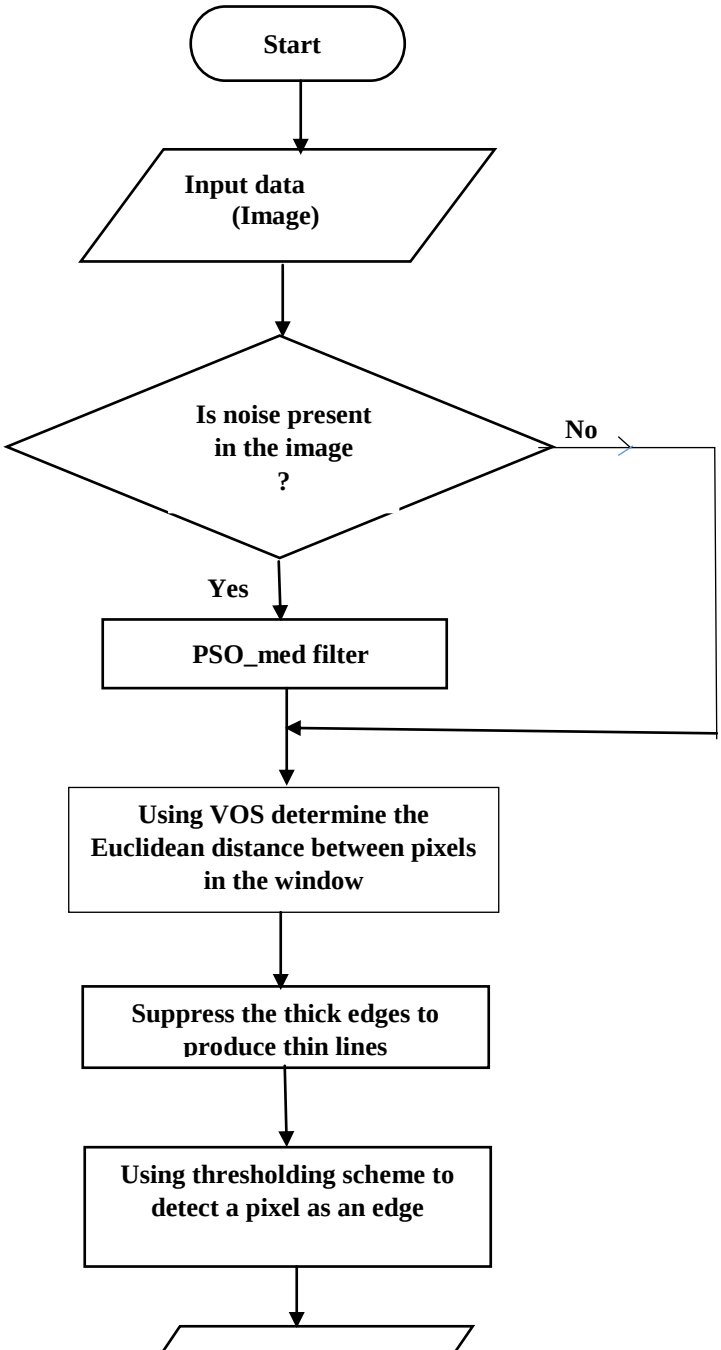

Start
Input data
(Image)
Is noise present
in the image
?
No
Yes
PSO_med filter
Using VOS determine the
Euclidean distance between pixels
in the window
Suppress the thick edges to
produce thin lines
Using thresholding scheme to
detect a pixel as an edge

Figure3.8:Flow Chart of the Proposed Edge Detection Algorithm

3.5 The Proposed Edge Detection Algorithm

The program function based on the following algorithms are as follows:

3.5.1 Particle swarm optimization algorithm

The algorithm is described as follows:

1. Input the image.

The benchmark images in Figure 2.18 are used as inputs.Theimagesare selected from the database of the computer system. The following program listing shows snippets of the portion that extracts the image from the database.Figure 3.9 shows one of the extracted images from the computer database.

```
filename = uigetfile('*.jpg;*.tif;*.png');
data=im2double(imread(filename));
```

Figure 3.9: Extracted Input Image form the Computer Database

2. The program checks if noise is present in the input image. If there exists noise in the image, the particle swarm optimization median filter is initialized. The following program listing shows snippets of the portion that initializes the particle swarm optimization median filter.

```
r1 = components(:,:,1);% first components of the image
g1 = components(:,:,2);% second component of the image
b1 = components(:,:,3);% third component of the image
r1 = medfilt2(r1,[3 3]);
g1 = medfilt2(g1,[3 3]);
b1 = medfilt2(b1,[3 3]);
result(:,:,1) = (r1);
result(:,:,2) = (g1);
result(:,:,3) = (b1);
output = result;
% initializing the particle swarm optimization
[BestPosition,bestAdaption,AdaptionArray] = PSO(5,10,result,output);
```

3. Initialize the position x(t) and velocity v(t) for the particle.
4. For every pixel in the population size, evaluate the fitness function.
5. Generate values for the weight, acceleration coefficient and the random values.
6. Update the position and velocity as in equations (2.10) and (2.11). The following program listing shows snippets of the portion that updates the position and velocity of the particle swarm optimization.

```
w=0.9-t*(0.8/iterator);
a1=pbest-p;
a2=gbest-p;
tempv=w.*v+1.4962*rand(1,1).*a1+1.4962*rand(1,1).*a2;
```

7. When f(present(t+1)) < f(Pbest(t)), update the Individual Best for i (particle), the set of weights that yields the (Best Fitness value) minimum MSE. The following program listing shows snippets of the portion that set the weights that yields the best fitness value.

```
fi=GetAdaption(p,noisy,output);
if fi>pMax
pMax=fi;
```

```
        pbest=p;
    end
```

8. When f(Gbest(t)) < f(present(t+1))), update the Global Best, the set of weights that yields the minimum MSE in a global sense (i.e.,) Best of Individual Best's. The following program listing shows snippets of the portion that set the weights that yields the best fitness value

```
if pMax>gMax
    gbest=pbest;
    gMax=pMax;
end
```

9. The algorithm is iterated until convergence is reached. This convergence yields Gbest (t), the optimal set of weights that minimizes the mean square error. With the set Gbest (t) as weights, the filter estimates the corrupted pixel.

3.5.2 Algorithm for the vector order statistics

The algorithm is described as follows:

10. Using the reconstructed image as an input image, generate a 3x3 pixel window from the image. The following program listing shows snippets of the portion that generates a 3x3 pixel window from the input image

```
A =(floor(result*255));% Image is an m-by-n-by-3 array of integers in the range [0, 255];
[maxrow, maxcol, depthA] = size(A);
DistanceX = maxrow-2; % the total dimension of the image in the x-direction minus 2 assumes a 3x3 window
DistanceY = maxcol-2; % the total dimension of the image in the y-direction minus assumes a 3x3 window
```

11. For each pixel in the window a vector of size 3 is used to describe the colour, this is written as $P_{p,q}$ RGB. The vector is the RGB values of that pixel. The following program listing shows snippets of the portion.

```
for row = 0: DistanceX-1
for column = 0: DistanceY-1
    row1 = row + 1;
    row2 = row + windowWidth;
    column1 = column + 1;
    column2 = column + windowHeight;
    iWindow = A(row1:row2, column1:column2, :); %produces a 3x3 window for data
```

12. A new set of scalars A_0 to A_8, is calculated for each pixel by determining the Euclidean distance between a given $P_{p,q}$ RGB and all other $P_{p,q}$ RGB in the window.The Euclidean distance between a pair of vector is the difference in each vector's R value squared added to the differences in G values squared added to the difference in B values squared and the sum in a square root. This result in a 9 distance which is A_0 to A_8.The following program listing shows snippets of the portion that calculates the Euclidean distance.

```
for row = 1: maxrow
for column = 1: maxcol
        Xi(1:maxrow,1:maxcol,1) = B(row,column,1);
        Xi(1:maxrow,1:maxcol,2) = B(row,column,2);
        Xi(1:maxrow,1:maxcol,3) = B(row,column,3);
    Di = abs(Xi - B);
    Di = Di.^2;
    Di = (sum(Di, 3)).^.5;
    d(row,column) = sum(sum(Di));
end
end
```

13. This process is repeated for each pixel in the window, resulting in a single scalar for each pixel. So, the initial 3x3x3 matrix has been transformed into a 3x3x1.
14. The new 3x3x1 matrix is reshaped into a 9x1 array where the index corresponds to the pixel number.
15. The original 3x3x3 window is reshaped in a manner related to the reshaping of the 3x3x1 matrix to result in a 9x3 matrix where the first dimension corresponds the same pixel number as the 3x3x1 array index, and the second dimension is the RGB values for that

pixel. The following program listing shows snippets of the portion that rearrange the set of pixels

```
d2 = reshape(d, [],1);
B2 = reshape(B, [],3);
[d2, I] = sort(d2);
B2sorted = B2(I, :);
```

16. The 9x1 array is sorted into ascending order, and the same rearrangement of indices is applied to the first dimension of the 9x3 matrix.
17. The minimum vector range is then calculated by finding the Euclidian distance between the first pixel and the last pixel resulting from the ordering of the sort.The following program listing shows snippets of the portion that calculates the minimum vector range between pixels.

```
for nVector = 1: k
   diffVect = Z(end + 1 - nVector,:) - Z(1,:);
   diffVect = diffVect.^2;
   diff(nVector) = sum(diffVect);
end
minVR = min(diff);
```

18. If the vector range is above a user set threshold then the window contains an edge, and the center pixel of the window is set to 1 to represent an edge at that location.Figure3.10 shows the output result of the algorithm before edge suppression.

Fig 3.10: Output of the Vector Order Statistics

19. Suppress the thick edges produced by the vector order statistics with a view to achieving thin and continuous edge lines. The following program listing shows snippets of the portion that suppress the thick edges to produce thin lines.

```
    [row,column] = size(Image);
    thin = zeros(row,column);
    clrimg = fspecial('average',[fsize,fsize]);
    ImSmooth = imfilter(Image,clrimg,'replicate');
    fy = [3 4 4;4,4 3;4 4 4 ];
    fx = [3 4 4;3 4,5;4 4 4];
    ax = imfilter(ImSmooth,fx,'replicate');
    ay = imfilter(ImSmooth,fy,'replicate');
    mag = sqrt((ax.*ax) + (ay.*ay));
    [m,n] = size(mag);
switch direction
case 1
idx = find((iy<=0 & ix>-iy)  | (iy>=0 & ix<-iy));
case 2
idx = find((ix>0 & -iy>=ix)  | (ix<0 & -iy<=ix));
case 3
idx = find((ix<=0 & ix>iy) | (ix>=0 & ix<iy));
case 4
idx = find((iy<0 & ix<=iy) | (iy>0 & ix>=iy));
end
```

The output result after suppressing the thick edges is depicted in Figure3.11

Figure3.11: Output Result of the Suppressed Edges

20. Calculate the Peak-Signal-Noise Ratio that determines the quality of the restored imageby using equations (2.12) and (2.13). The following program listing shows snippets of the portion that generates the value for the Peak-Signal-Noise Ratio.

```
[m0,n0]=size(output);
[m1,n1]=size(data2);
m=min(m0,m1);
```

```
n=min(n0,n1);
 x=0;
z=m*n;
for h=1:m
     for j=1:n
     x=x+(double(data2(h,j))-double(output(h,j)))^2;
     end:
end
mse=x/z;
maxium=255*255;
Psnr=10*log10(maxium/mse);
```

CHAPTER FOUR

RESULTS AND DISCUSSIONS

4.1 Introduction

This chapter discusses the result obtained from the Berkeley benchmark databases, facial and satellite images of high resolution in comparison with existing proven traditional edge detection algorithms.

4.2 Analysis of the Proposed Edge Detection Algorithm

The proposed algorithm was tested on benchmark images obtained from the Berkeley image database, remotely sensed images obtained from Google earth software and sample of natural faces. The images obtained were corrupted with impulsive noise (salt & pepper noise) at different noise levels at interval of 10% to 90% with a view to testing the effectiveness of the proposed algorithm in noisy environment. An image corrupted with 10% noise simply means that 10% of the total pixels that constitutean image is affected by noise. The proposed algorithm

was based on collection of pixels on an image rather than just a single pixel and its neighbor as applied in existing traditional edge detection algorithms. The corrupted noisy image was used as an input and passed through the particle swarm optimization algorithm for reconstruction using Appendix A1.

The particle swarm optimization algorithm used the following parameters: w = 0.81, C_1 = 1.4962 and C_2 = 1.4962. The inertia weight w is set with a view to achieving convergence and the acceleration coefficient C_1 and C_2 are set to equal integer values with a view to improving performance. The output image obtained from the particle swarm optimization algorithm was then passed into the vector order statistic algorithm using Appendix A2. The vector order statistics algorithm selected a 3x3 window size from the output image obtained from the particle swarm optimization algorithm with a view to reducing the computation time of the algorithm.The output image from the vector order statistics algorithm is now suppressed to reduce the thick edges in the output image using Appendix A3. Figure4.1 and Figure4.2 show the result of applying the particle swarm optimization based on vector order statistics algorithm on an image respectively.

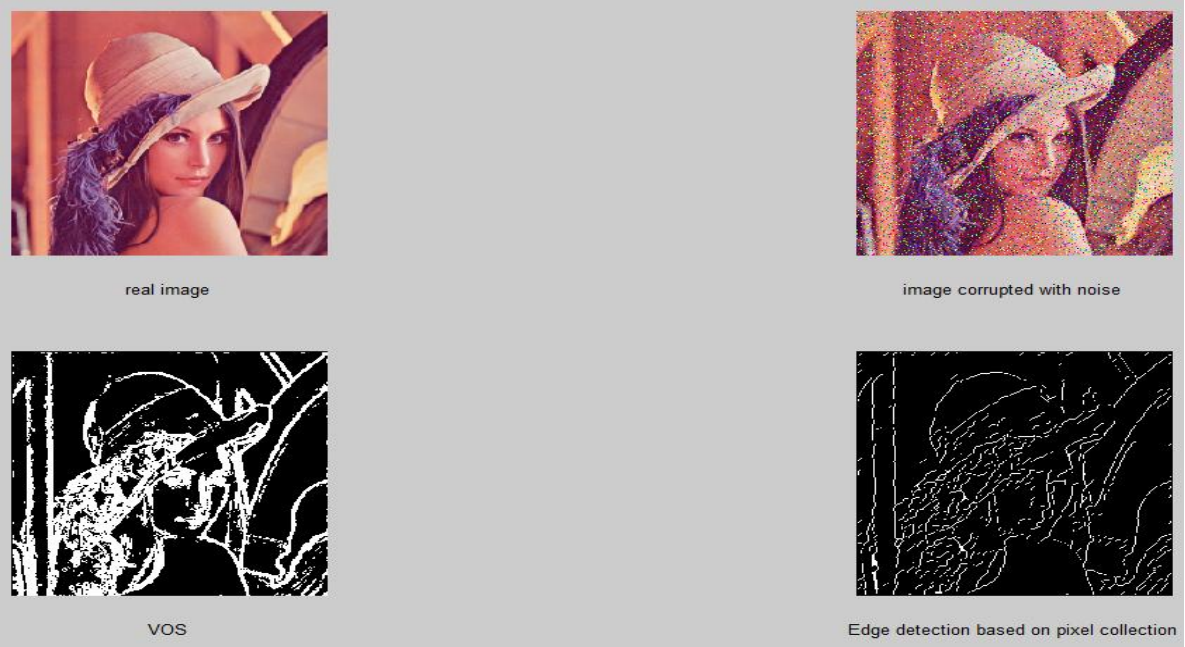


Figure4.1: Output Results on Test Image (1)

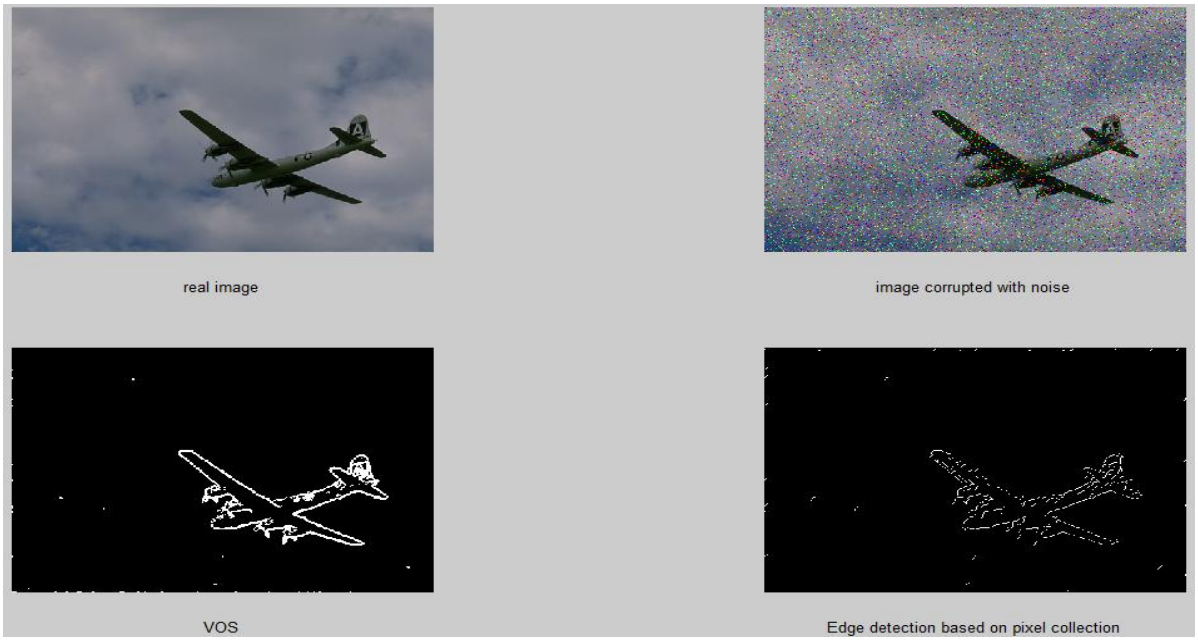


Figure4.2: Output Results on Test Image (2)

Four images in the output result are displayed in Figures 4.1 and 4.2 The first image represents the benchmark image before it was corrupted with noise, the second image represents the

corrupted image, the third image represents the output of the vector order statistics and the fourth image represents the final output image obtained based on collection of pixels.The peak-signal-noise ratio at different noise levels is shown in Table 4.1 obtained using Appendix A2 as a measure of quality between the original image and the reconstructed image. The higher the peak signal-to- noise ratio, the better the quality of the reconstructed image.Theproposedalgorithm performedin clean and noisy environment of noise level 33% and below.The peak-signal-noise ratio at noise level of 33% was found to be 57.7320dB.If the noise level in the image is above 33%, the algorithm becomes sensitive to noise. The summary of peak to signal noise ratio is shown in Table 4.1

Table 4.1: Noise Levels with their Respective Peak Signal-to-Noise Ratio

Percentage pixel affected by noise (%)	PSNR (dB)
10	62.7104
20	59.8679
30	58.1529
40	56.9401
50	54.9414
60	54.1182
70	53.3830
80	53.7189
90	53.0920

The plot of peak-to-signal noise ratio is shown in Figure4.3 for the various noise levels.

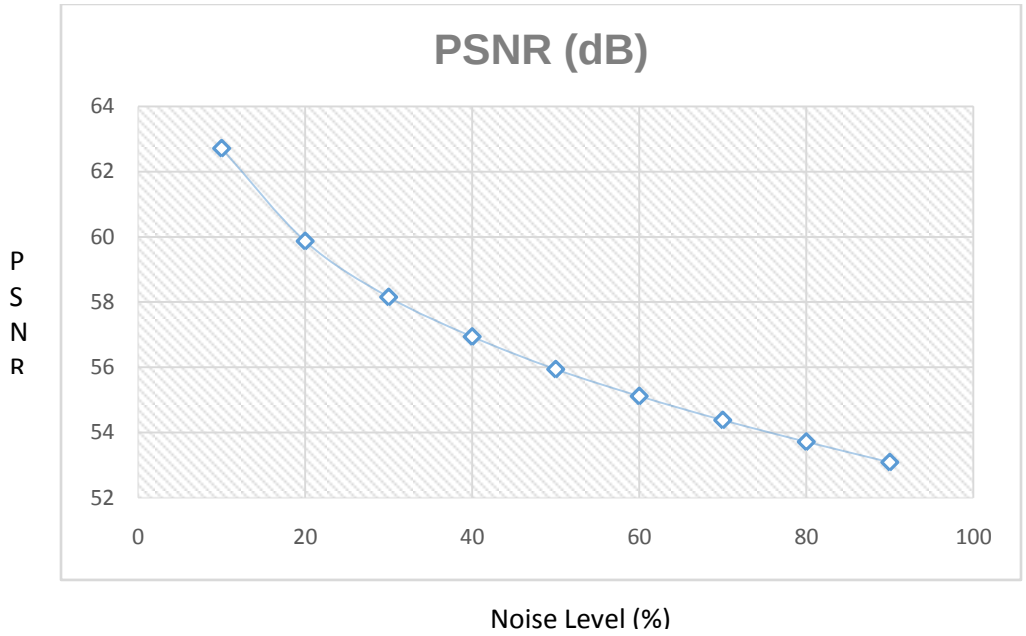


Figure4.3:Peak Signal to Noise Ratio for Various Noise Levels

Figure4.3 shows that, as the noise level in the images increases, the peak signal-to-noise ratio decreases. This implies that the quality of the reconstructed image decreases with increase in noise level.

4.1.1 Computation time

Computation time or running time is the total length of time required to perform a computational process. The individual time required to run each step in the algorithm is presented in Table 4.2. The algorithm is sub-divided into three steps which are shown in Appendix A1, Appendix A2 and Appendix A3. The overall computation time to run the algorithm is the sum of individual times used for each of the steps.

Table 4.2: ComputationTime

Steps	Consumed time (sec)

Particle Swarm Optimization algorithm	99.798
Vector Order Statistic Algorithm	30.222
Reduced thick edges	9

The computation time of the algorithm depends on the type of image to be processed (dimension of the image), the level of noise present in the images and the specification of the computer system running the program. It is also dependent on the number of iterations and population size in the particle swarm optimization algorithm. The specification of the system used to run the algorithm are as follows:

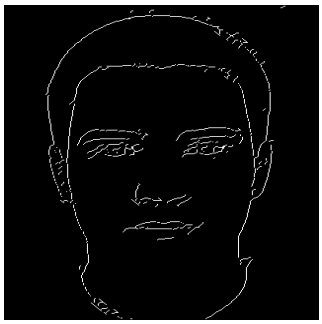
Processor: Intel ® Core ™ i3-2350M CPU @ 2.30GHz

RAM: 4GB

System type: x64-based processor

The sample of natural faces collected in Figure 2.20are used as the input for the edge detection algorithm and the results are shown in Figure 4.4.

Facial Image (1) Facial Image (2)



Facial Image (3)

Figure 4.4:Output Result of Applying the ProposedEdge Detection Algorithm on Facial Images

Figure 4.4 shows the edge maps of applying the particle swarm optimization algorithm based on vector order statistics to the sample of collected natural faces used. The visual performance of the result of applying the algorithm showed that the edges that constitute the overall face was extracted completely. The algorithm is not dependent on the type of skin colour in the image.Sharp and high resolution images as in facial images (2) and (3) showed clear extracted edge maps of the overall face than low resolution image as in facial image (1).But with high resolution images the processing time increases, unless if used on a dedicated and specialized systems. But the most critical issue affecting this edge detection algorithm in facial image detection is facial expression. The same individual with two or more expressions can look entirely different after extraction of edges that constitute the face.

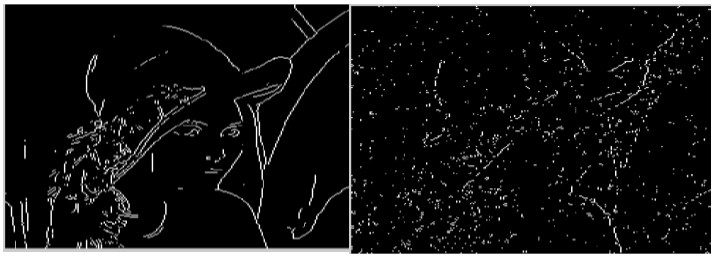
The result of applying the proposed edge detection algorithm to the sample of the remotely sensed image in Figure 2.21 is presented in Figure 4.5.

Figure 4.5: Output Result of Applying the ProposedEdge Detection Algorithm to Remotely Sensed Images.

Figure 4.5 shows the edge maps of applying the particle swarm optimization algorithm based on vector order statistics to the sample of remotely sensed image used. The visual performance of the result of applying the algorithm showed extracted regions and roads in the remotely sensed image.

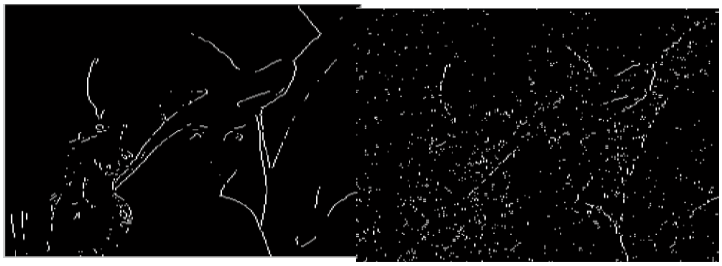
The benchmark image in Figure 2.18 was used to compare the proposed algorithm in noisy and clean environment with the existing proven edge detection algorithms. The output results are presented in Figures 4.6, 4.7, 4.8, 4.9, 4.10and4.11

The Figures 4.6(a)–4.11(a) are the output results in clean environment and the Figures 4.6(b)–4.11(b) are the output results in noisy environment.



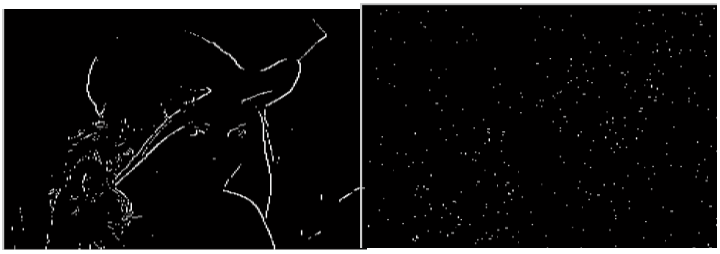
(a) Clean(b) Noisy

Figure 4.6: Sobel Edge Detection Algorithm



(a) Clean (b) Noisy

Figure 4.7: Prewitt Edge Detection Algorithm

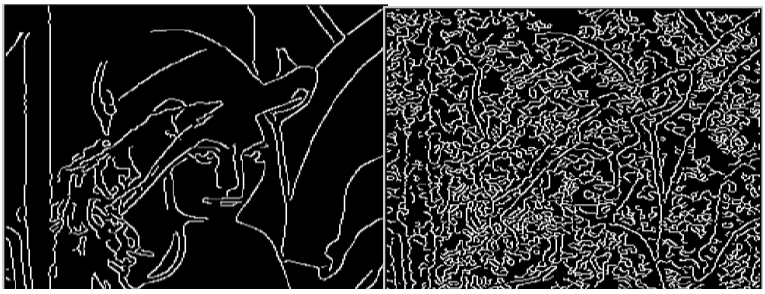


(a) Clean (b) Noisy

Figure4.8: Roberts Edge Detection Algorithm

(a) Clean(b) Noisy

Figure4.9: Laplacian Edge Detection Algorithm



(a) Clean (b) Noisy

Figure4.10: Canny Edge detection Algorithm

(a) Clean(b) Noisy

Figure4.11: Proposed Edge Detection Algorithm

In the work of (Dutta & Chaudhuri, 2009) the algorithm in the RGB colour space was tested on simple square shape depicted in Figure 4.12. The work of (Almadhoun, 2013) was an improvement on the (Dutta & Chaudhuri, 2009) which also tested the algorithm on Figure 4.12 with a view to comparing results visually.

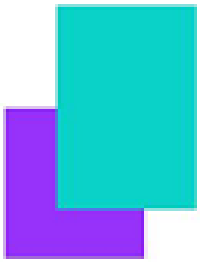
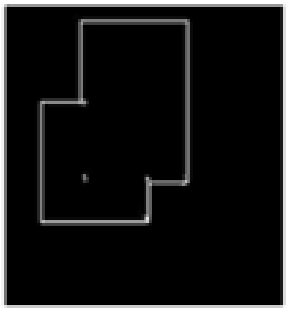


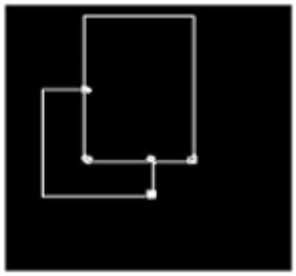
Figure 4.12: Simple Shape Used to Test the Algorithm of (Dutta & Chaudhuri, 2009) and (Almadhoun, 2013)

The output result of the two algorithms in visual comparison with the proposed particle swarm optimization algorithm based on vector order statistics is presented in Figure 4.13. The output in Figure 4.13(a) is the result of applying the algorithm developed by (Dutta & Chaudhuri, 2009) to the shape in Figure 4.12 and the output in Figure 4.13(b) is the result of applying the developed algorithm by (Almadhoun, 2013) to the shape in Figure 4.12. Also the output in

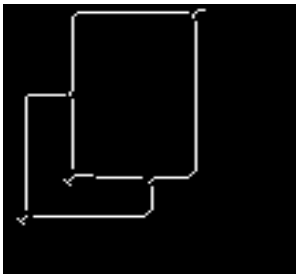
Figure 4.13(c) is the result of applying the proposed particle swarm optimization algorithm based on vector order statistics to the simple shape in Figure 4.12. The output result of the work of (Dutta & Chaudhuri, 2009) showed missing edges while the output result of the work of (Almadhoun, 2013) showed thick edges at joints as compared with the proposed algorithm.



(a) Output Edge Map with Missing Edges (Dutta & Chaudhuri, 2009)



(b) Output Edge Map with Jointed Thick Edges (Almadhoun, 2013)



(c) Output Edge Map Produced Using the Proposed Algorithm

Figure 4.13: Comparison of Images Produced by Different Algorithms on Sample Shape

The work of (Bhardwaj & Mann, 2013) used the Adaptive Neuro-Fuzzy Inference system (ANFIS) to detect edges in noisy images. The output result of applying the developed algorithm in visual comparison with the proposed particle swarm optimization based on vector order statistics is presented in Figure 4.14. Figure 4.14(a) is the output result of the work of (Bhardwaj

& Mann, 2013) and Figure 4.14(b) is the output result of the proposed particle swarm optimization based on vector order statistics algorithm.

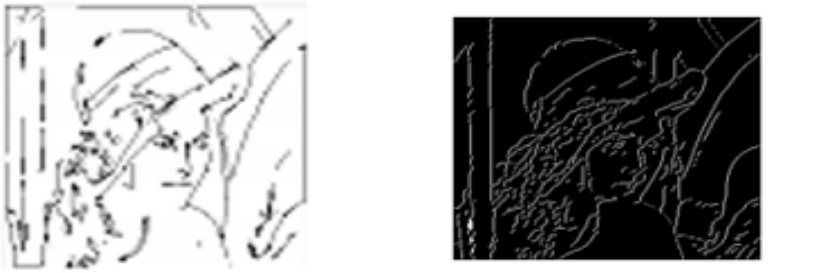
(a) Output Edge Map Produced by (Bhardwaj *et al.*, 2013)

(b) Output Edge Map Produced by the Proposed Algorithm

Figure 4.14: Output Results of the Proposed Algorithm in Visual Comparison with the Work of (Bhardwaj & Mann, 2013) in Noisy Environment

The output result in Figure 4.14 is a visual comparison between the proposed algorithm and the work of (Bhardwaj & Mann, 2013). The generated output edge map in Figure 4.14(a) produces more broken and missing edges as compared to the proposed algorithmin Figure 4.14(b).

The work of (Rai & Dutta, 2013) used the ant colony optimization technique to detect edges in images. The result of applying the algorithm and the proposed algorithm is presented in Figure 4.15 as a means of visual comparison.Figure 4.15(a) is the output result of the work of (Rai & Dutta, 2013)and Figure 4.15(b) is the output result of the proposedalgorithm.



(a) Output Edge Map Produced by (Rai & Dutta, 2013)

(b) Output Edge Map Produced by the Proposed Algorithm

Figure 4.15: Output Results of the Proposed Algorithm in Visual comparison with the work of (Rai & Dutta, 2013) in clean environment

The output result in Figure 4.15in visual comparison with the proposed algorithm showed that the work of (Rai & Dutta, 2013) as in Figure 4.15(a) produces blurred and broken edges as compared with the proposed algorithmas in Figure 4.15(b).

Table 4.3 shows the Pratt Figure of Merit for the various edge detection algorithms as a means of validation. The values for the Pratt Figure of Merit was obtained using equation (2.25)

Table 4.3: Pratt Figure of Merit for Various Edge Detection Algorithm

Edge detection algorithm	Image without noise	Image with noise
Sobel edge detection algorithm	0.4209	0.4191
Prewitt edge detection algorithm	0.4195	0.4191
Robert edge detection algorithm	0.4181	0.2807
Laplacian edge detection algorithm	0.7048	0.2811
Canny edge detection algorithm	0.8472	0.5606
Proposed edge detection algorithm	0.8480	0.8458

Figure 4.16 shows the graphical representation of the Pratt Figure of Merit (PFOM) in comparison with the existing traditional edge detection algorithm such as

SEDA = Sobel Edge Detection Algorithm PEDA = Prewitt Edge Detection Algorithm

REDA = Roberts Edge Detection Algorithm LEDA = Laplacian Edge Detection Algorithm

CEDA = Canny Edge Detection Algorithm DEDA = Proposed Edge Detection Algorithm

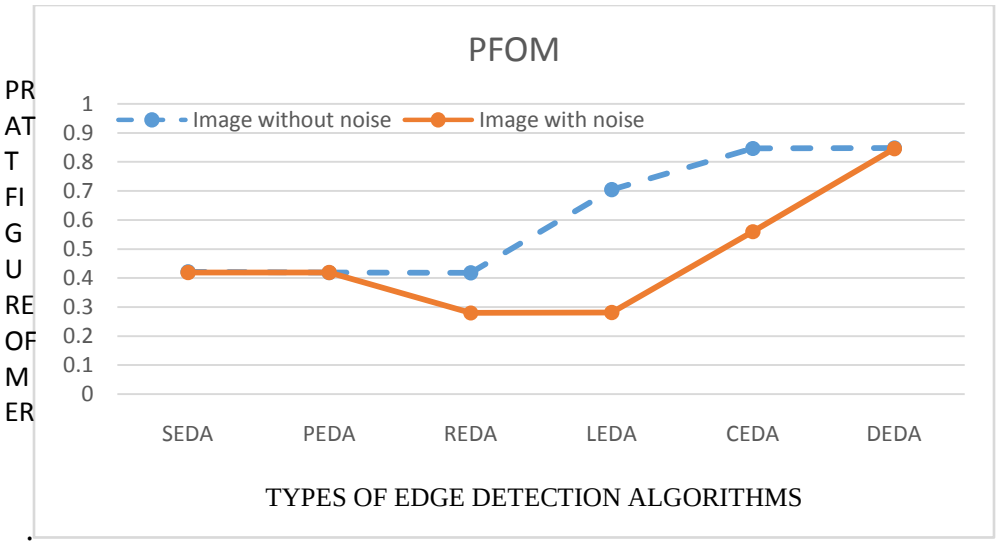


Figure 4.16: Quantitative Comparison Using Pratt Figure of Merit (PFOM)

The plot for the Pratt Figure of Merit (PFOM) showed that in the absence of noise, the values obtained are closer to the value 1 than the values obtained when noise is present in the image for the existing proven edge detection algorithms. The PFOM value obtained for the proposed edge detection algorithm in clean and noisy environment showed lessermargin than those of the existing proven edge detection algorithm. Theseresults signify that the proposed algorithm performed better than the existing proven edge detection algorithms.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 Introduction

This chapter presents a summary of the research work, thelimitationsthat were observed during the research work, the recommendations being offeredand the conclusion.

5.2 Summary

This research work presents an improved edge detection algorithm using particle swarm optimization based on vector order statistics. In order to address the shortcomings of the existing traditional edge detection algorithms that could not process coloured images directly unless been converted to gray scale, Vector Order Statistic technique was employed. The profile edge intensity was applied to generate a collection scheme. This collection scheme was obtained using set of pixels with respect to the step and roof edge profiles. The collection scheme was thenapplied as a mask in both vertical and horizontal directions to the images.An improved figure of 0.8480 and 0.8458 were obtained in clean and noisy environment respectively using the PFOM. The proposed algorithm achieved a PSNR of 57.7320dB in environment containing ≤ 33% of noise level.

5.3 Conclusions

The development of the proposed algorithm which is a Particle Swarm Optimization technique based on Vector Order Statistics for detection of edges in noisy coloured images is presented in this research work. The algorithm was implemented in Matlab 2013 image processing toolbox.The proposed algorithm was tested on Berkeley benchmark images with a view to obtaining both visual and quantitative comparison with the existing proven edge detection algorithms. The proposed algorithm was also tested on facial and satellite real life images. Based on the result obtained in visual and quantitative comparison with the proven edge detection algorithms, it could be concluded that the pixels that constitute the edges in an image have been extracted completely. Also, a noticeable performance in noisy environment in comparison with the proven edge detection algorithms was recorded. However, the proposed algorithm can perform only in noisy environment of noise level ≤ 33%. The result obtained from the Pratt

Figure of Merit for clean and noisy environment using the proposedalgorithm are 0.8480 and 0.8458, respectively. This result signifies 3% improvement in detection of edges in noisy environment as compared with the proven existing traditional edge detection algorithms.

5.4 Limitations

During the course of this research work, certain limitations were observed which are itemized as follows:

i The optimal performance of the algorithm depended on the threshold value. The threshold values were adjusted manually to get a desired result.

ii The proposed algorithm could only perform in noisy environment of $\leq 33\%$

iii The scope of the work was limited to edge detection in colour images, hence region based segmentation and feature extraction was not captured

5.5 Recommendations and Further Work

Future works should consider the following areas:

i. The algorithm can be extended to detect edges in images of higher noise levels.

ii. The algorithm can be extended to region based segmentation and implementation on hardware

iii. The algorithm can be developedto use automatic thresholding value for edge detection

REFERENCES

Abdul A., & Funjan W. (2013). Denoising An Image Based On Particle Swarm Optimization (PSO) Algorithm. *journal of Babylon University/Pure and Applied Sciences, 21*(5), pp.1511-1518.

Akram A., & Ismail A. (2013). Comparison of Edge Detectors. *International Journal of Computer Science and Information Technology Research (IJCSITR), 1*(1), pp.16-24.

Almadhoun M. D. (2013). Improving and measuring color edge detection algorithm in RGB color space. *International Journal of Digital Information and Wireless Communications, 3*(1), pp.19-24.

Anila, & Devarajan. (2010). Simple and Fast Face Detection System Based on Edges. *International Journal of Universal Computer Sciences, 1*(2), pp.54-58.

Ashour A. S., El-Sayed M. A., Waheed S. E., & Abdel-Khalek S. (2014). New Method Based on Multi-Threshold of Edges Detection in Digital Images. *International Journal of Advanced Computer Science & Applications, 5*(2), pp.90-99.

Bhardwaj K., & Mann P. S. (2013). Edge Detection in Digital Images Corrupted by Salt and Pepper Noise using Adaptive NeuroFuzzy Inference System(ANFIS). *International Journal of Computer Applications, 72*(12), pp.36-41.

Blondin J. (2009). Particle Swarm Optimization: A Tutorial. *retrieved from www.cs.armstrong.edu/saad/csci8100/pso_tutorial.pdf, on 23/11/2014*, pp.1-5.

Chaudhary A., & Gulati T. (2013). segmenting digital images using edge detection. *International Journal of Application or Innovation in Engineering & Management (IJAIEM), 2*(5), pp.319-323.

Chen X., & Chen H. (2010). A Novel Color Edge Detection Algorithm in RGB Color Space *Institute of Electrical and Electronics Engineers Transcations*, pp.793-796.

Dutta S., & Chaudhuri B. (2009). A Color Edge Detection Algorithm in RGB Color Space. *IEEE-International Conference on Advances in Recent Technologies in Communication and Computing*, pp.337-340.

Eberhart R., & Shi Y. (2001). Particle Swarm Optimization: Developement, Applications and Resources *Institute of Electrical and Electronics Engineers Transcations (IEEE), 3*(1), pp.81-86.

Gang L., Fan Y., & Linlin W. (2008). An Algorithm for Remote Sensing Image Edge Detection Based on Fuzzy Sets. *Second International Symposium on Intelligent Information Technology Application*, pp.1087-1090.

Ghasemi M., Koohi M., & Shakery A. (2011). Edge detection in multispectral images based on structural elements. *The International Journal of Multimedia & Its Applications (IJMA), Vol.3*(No.1), pp.90-99.

Gonzalez R., & Woods R. (2008). *Digital Image Processing* I. Pearson Education (Ed.)(pp. 68-122).

Gorai A., & Gosh A. (2009). Gray Level Image Enhancement By Particle Swarm Optimization. *Institute of Electrical and Electronics Engineers Transcations*, pp.72-77.

Gudipalli A., & Tirumala R. (2013). Comprehensive Edge Detection Algorithm for Satellite Images. *World Applied Sciences Journal, 28*(8), pp.1042-1047.

Gupta S., & Mazumdar S. G. (2013). Sobel edge detection algorithm. *International journal of computer science and management Research, 2*(2), pp.1578-1583.

Haque M. J., & Aljahdali S. H. (2013). An Approach for Reconstructed Color Image Segmentation using Edge Detection and Threshold Methods. *International Journal of Computer Applications, 68*(11), pp.32-36.

Jahanzeb S., & Siddiqui A. (2013). Analysis of Edge Detection Algorithms for Feature Extraction in Satellite Images *IEEE International Conference on Space Science and Communication (IconSpace), 3*(1), pp.238-242.

Jothilakshmi R., & Rajeswari R. (2014). Modified Ant Colony Optimization Based Approach for Edge Detection in Images. *International Journal of Engineering Research and Technology, 3*(1), pp.3384-3389.

Katiyar, & Arun. (2014). Comparative analysis of common edge detection techniques in context of object extraction. *IEEE Transactions of Geoscience and Remote Sensing, 50*(11), pp.68-79.

Kaur A., & Singh M. D. (2012). An Overview of PSO- Based Approaches in Image Segmentation *international journal of engineering and technology, 2*(8), pp.1349-1357.

Kaur B., Garg A., & Kaur A. (2010). Mathematical Morphological Edge Detection For Remote Sensing Images. *International Journal of Electronics & Communication Technology (IJECT), 1*(1), pp.29-33.

Koschan A., & Abidi M. (2005). Detection and classification of edges in colour images. *IEEE Signal processing magazine, special issue on colour image processing, 22*(1), pp.64-75.

Kyprianidis J. E., Kang H., & Döllner J. (2009). Image and Video Abstraction by Anisotropic Kuwahara Filtering. *retrieved from www.kyprianidis.com/p/npar2009/jkyprian-npar2009.pdf, on 04/08/2014*, pp.1-9.

Maini R., & Aggarwal H. (2011). Study And Comparism of Various Image Edge Detection Technique. *internation Journal of Image Processing (IJIP), 3*(1), pp.1-12.

Ndajah P., Kikuchi H., & Yukawa M. (2011). An Investigation on The Quality of Denoised Images. *International Journal of Circuits, Systems and Signal Processing, 5*(4), pp.423-434.

Neupane B., Aung Z., & Woon W. L. (2012). A New Image Edge Detection Method Using Quality-Based Clustering. *Proceedings of the IASTED International Conference Visualization, Imaging and Image Processing, July 3-5*(Retrieved from www.dnagroup.org/techreps/DNA-2012-01.pdf on 02/10/2014), pp.20-26.

Pande S., Bhadouria V. S., & Ghoshal D. (2012). A study on edge marking scheme of various standard edge detectors. *International Journal of Computer Applications, 44*(9), pp.33-37.

Panetta K., & Wharton E. J. (2008). Logarithmic Edge Detection with Applications *Journal of Computers, 3*(9), pp.11-19.

Petrou M., & Bosdogianni P. (1999). *Image Processing the Fundermentals* (pp. 1-325).

Pham T. (2010). Non-maximum Suppression Using fewer than Two Comparisons per Pixel. *Canon Information Systems Research Australia (CiSRA)*, pp.1-11.

Rai P., & Dutta M. (2013). Image Edge Detection Using Modified Ant Colony Optimization Algorithm Based on Weighted Heuristics. *International Journal of Computer Applications, volume 68*(issue 15), pp.5-9.

Rajasekaran A., & Senthilkumar. (2014). Image Denoising Using Median Filter with Edge Detection Using Canny Operator. *International Journal of Science and Research, 3*(2), pp.30-34.

Rashmi, Kumar M., & Saxena R. (2013). Algorithm and Technique on Various Edge Detection: A survey. *Signal & Image processing : An International Journal (SIPIJ), 4*, pp.65-75.

Roomi M., & Rajee J. (2011). Speckle Removal in Ultrasound Images Using Particle Swarm Optimization Technique. *IEEE-international conference on recent trends in information technology, Retrieved from www.ieeexplore.ieee.org on 21/03/2014*, pp.926-931.

Roomi M. M., & Karuppi M. (2010). A Particle Swarm Optimization Based Edge Preserving Impulse Noise Filter *Journal of Computer Science, 9*(6), pp.1014-1020.

Saini V., & Garg R. (2012). A Comparative Analysis on Edge Detection Techniques Used in Image Processing *IOSR Journal of Electronics and Communication Engineering, 1*(2), pp.56-59.

Setayesh M., Mengjie, & Johnston M. (2013). A novel particle swarm optimization approach to detecting continuous, thin and smooth edges in noisy images. *Elsevier Inc.*, pp.28-51.

Shi Y. (2004). Particle Swarm Optimization. *institute of Electrical and Electronics Engineers Transcations (IEEE) Neural Network Society*, pp.8-13.

Solomon C., & Breckon T. (2011). Fundermentals of Digital Image Processing A Practical Approach With Examples in MATLAB (First ed., pp. 1-109): John Wiley & Sons Ltd.

Sukhendu D. (2013). Concepts in Edge detection. *retrieved from www.cse.iitm.ac.in/~vplab/courses/CV_DIP/.../LECT_EDGE_DET.pdf on 03/01/2014*, pp.1-32.

Trahanias P. E., & Venetsanopoulos A. N. (1996). Vector order statistics operators as color edge detectors. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 26*(1), pp.135-143.

Venkata, & Babu J. (2012). Color Image Enhancement Using Praticle Swarm Optimization. *Internation Journal of Engineering Science and Technology (IJEST), 2*(3), pp.474-480.

Verma R., & Ali J. (2013). A Comparative Study of Various Types of Image Noise and Efficient Noise Removal Techniques. *International Journal of Advanced Research in Computer Science and Software Engineering, 3*(10), 617-622.

Vijayarani, & Vinupriya. (2013). Performance Analysis of Canny and Sobel Edge Detection Algorithms in Image Mining. *International Journal of Innovative Research in Computer and Communication Engineering, 1*(8), pp.1760-1767.

Wesolkowski S., & Jernigan E. (1999). Color edge detection in RGB using jointly euclidean distance and vector angle. *retrieved from http://www.cipprs.org/papers/VI/VI1999/pp009-016-Wesolkowski-Jernigan-1999.pdf., on 10/02/2014*, pp.9-16.

Xin G., Ke C., & Xiaoguag H. (2012). An improved Canny edge detection algorithm for color image. *Institute of Electrical and Electronics Engineers Transcations*, pp.113-117.

Xiong Y. (2014). Research on an Edge Detection Algorithm of Remote Sensing Image Based on Wavelet Enhancement and Morphology *Journal of Computers, 9*(5), pp.1247-1252.

Zhen-Gang S., & Qin-Zi L. (2010). Edge detection for Medical Image based on PSO Algorithm. *IEEE Third International Conference on Intelligent Networks and Intelligent Systems, Retrieved from www.ieeexplore.ieee.org on15/04/2014*, pp.84-87.

AppendixA1

M-file code (main file)

```
clc;

clear;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SADIQ BASHIR OLANIYI MSC/ENG/21179/2012-2013
%A PSO edge detection algorithm based on vector order statistics. The data
%component in the image are processed as a vector, finding the Euclidean
%Distance between the pixels thus reducing the problem of processing the
%data channel independently and using a fusion technique to join them.
%A collection scheme is proposed for collection of pixels based on the step
and ramp edges with a view to reducing false and broken edges in theimage.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
filename = uigetfile('*.tif,*.jpg');

data=im2double(imread(filename));
```

```
i=input('please input the percentage of Noise=');
noisy= imnoise(data,'salt & pepper',i);
r=noisy(:,:,1);
g = noisy(:,:,2);
b = noisy(:,:,3);
components(:,:,1) = r;
components(:,:,2) = g;
components(:,:,3) = b;
r1 = components(:,:,1);% first components of the image
g1 = components(:,:,2);% second component of the image
b1 = components(:,:,3);% third component of the image
r1 = medfilt2(r1,[3 3]);
g1 = medfilt2(g1,[3 3]);
b1 = medfilt2(b1,[3 3]);
result(:,:,1) = (r1);
result(:,:,2) = (g1);
result(:,:,3) = (b1);
output = noisy;
% initializing the particle awarm optimization
[BestPosition,bestAdaption,AdaptionArray] = PSO(5,10,noisy,output)
b=10*BestPosition;
n=round(b);
n;
imagestructure=zeros(n); %the array of structuring element
shadata1=size(imagestructure);
imagestructurestart=1; %the start point of structuring element
imagestructureend=shadata1(2); %the end point of structuring element
dimension=size(noisy);
imageheight=dimension(1); % Image height in pixels
imagewidth=dimension(2); %Image width in pixel
data1=Mero(noisy,imagestructure,imagewidth,imageheight,imagestructurestart,im
agestructureend);
inputdata=data1;
dimension=size(inputdata);
imageheight=dimension(1);
imagewidth=dimension(2);
data2=Mdila(inputdata,imagestructure,imagewidth,imageheight,imagestructuresta
rt,imagestructureend);
```

```
Med = result;
subplot(2,2,2)
imshow(noisy)
xlabel('image corrupted with noise')
subplot(2,2,1)
imshow(data)
xlabel('real image')
high = [0.15,0.4];
low = [0.01,0.1];
A =(floor(result*255));% Image is anm-by-n-by-3 array of integers in the
range [0, 255];
[maxrow, maxcol, depthA] = size(A);
DistanceX = maxrow-2; % The total dimension of the image in the x-direction
minus 2 assumes a 3x3 window
DistanceY = maxcol-2; % The total dimension of the image in the y-direction
minus 2 assumes a 3x3 window
windowWidth = 3;
windowHeight = 3;
nTopVectors = 2;
MVRthresh =400;
for row = 0: DistanceX-1 %since the image is a unit8 tydata1, interger values
will range from [0 to DistanceY-1]
for column = 0: DistanceY-1 %since the image is a unit8 tydata1, interger
values will range from [0 to DistanceY-1]
        row1 = row + 1;
        row2 = row + windowWidth;
        column1 = column + 1;
        column2 = column + windowHeight;
        iWindow = A(row1:row2, column1:column2, :); %produces a 3x3 window
for data channel component
        Z = EuclideanDistSort(iWindow);
        MVR(row+1,column+1) = minVR(Z, nTopVectors);
end
end
EdgeImMVR = zeros(maxrow-2, maxcol-2);
%NOTE: Using a 3x3 window
for row = 1: DistanceX
for column = 1: DistanceY
```

```
if MVR(row, column) > MVRthresh;
            EdgeImMVR(row+1, column+1) = 1;
else
            EdgeImMVR(row+1, column+1) = 0;
end
end
end
A = (EdgeImMVR);
subplot(2,2,3)
imshow(A)
xlabel('VOS')
B3 = Sup_thick(A,3,low(1),high(1));
subplot(2,2,4)
imshow(B3)
xlabel('Edge detection based on pixel collection')
Psnr = evaluate(data2,output);
disp('display the peak to signal noise level in dB')
Psnr
data2=Med;
Psnr = evaluate(data2,output);
disp('display the peak to signal noise level in dB')
Psnr
function B2sorted = EuclideanDistSort(image)
[maxrow, maxcol, depth] = size(image);
B = double(image);
for row = 1: maxrow %since the image is a unit8 type, interger values will
range from [1 to maxrow]
for column = 1: maxcol
        Xi(1:maxrow,1:maxcol,1) = B(row,column,1); % B(:,:,1) denotes the RED
channel
        Xi(1:maxrow,1:maxcol,2) = B(row,column,2); % B(:,:,2) denotes the
GREEN channel
        Xi(1:maxrow,1:maxcol,3) = B(row,column,3); % B(:,:,3) denotes the
BLUE channel
        Di = abs(Xi - B);
        Di = Di.^2;
        Di = (sum(Di, 3)).^.5;
        d(row,column) = sum(sum(Di));
```

```
end
end
d2 = reshape(d, [],1); % reshape(X,...,[],...) calculates the length of the
dimension
%represented by [], such that the product of the dimensions equals
PROD(SIZE(d))
B2 = reshape(B, [],3);
[d2, I] = sort(d2);
B2sorted = B2(I, :);
end
function minVR = minVR(Z, k)
%minVR = minVR(X, k)
%Z is sorted vectors of image; k is how many top vectors to consider
for nVector = 1: k
    diffVect = Z(end + 1 - nVector,:) - Z(1,:);
    diffVect = diffVect.^2;
    diff(nVector) = sum(diffVect);
end
minVR = min(diff);
end
```

Appendix A2

```
function [BestPosition,bestAdaption,AdaptionArray] =
PSO(ParticalNum,iterator,noisy,output)
% initialise paritcle swarm optimization
pbest=0;
gbest=0;
gMax=0;
pMaxp = zeros(ParticalNum,1);
pbestp = zeros(ParticalNum,1);
pp = zeros(ParticalNum,1);
vp = zeros(ParticalNum,1);
AdaptionArray = zeros(ParticalNum,1);
```

```
for i=1:ParticalNum
    t= 1 ;
    p=rand(1,1);
    v=0;
    pMax=0;
    fi=GetAdaption(p,noisy,output);
if fi>pMax
        pMax=fi;
        pbest=p;
end
if pMax>gMax
        gbest=pbest;
        gMax=pMax;
end
    [v]=newv(1,v,p,pbest,gbest,iterator);
    [p]=newpp(p,v);
    pMaxp(i,:)=pMax;
    pbestp(i,:)=pbest;
    pp(i,:)=p;
    vp(i,:)=v;
    AdaptionArray(i,:) = gMax;
end
for t=2:iterator
for i=1:ParticalNum


      p=pp(i,:);
      v=vp(i,:);
      pMax=pMaxp(i,:);
      pbest=pbestp(i,:);
      fi=GetAdaption(p,noisy,output);
if fi>pMax
          pMax=fi;
          pbest=p;
end

      [v]=newv(t,v,p,pbest,gbest,iterator);
      [p]=newpp(p,v);

if pMax>gMax
          gbest=pbest;
          gMax=pMax;
```

```
end

        pp(i,:)=p;
        vp(i,:)=v;
        pMaxp(i,:)=pMax;
        pbestp(i,:)=pbest;
end

    AdaptionArray(i,:) = gMax;

end

BestPosition = gbest;
bestAdaption = gMax;
AdaptionArray = AdaptionArray;


function fi=GetAdaption(p,noisy,output)
a=10*p;
n=round(a);
imagestructure=zeros(n);
shadata1=size(imagestructure);
imagestructurestart=1;
imagestructureend=shadata1(2);
dimension=size(noisy);
imageheight=dimension(1);
imagewidth=dimension(2);
data1=Mero(noisy,imagestructure,imagewidth,imageheight,imagestructurestart,im
agestructureend);
data2=data1;
Psnr=evaluate(data2,output);
fi=Psnr;
function
data2=Mdila(inputdata,imagestructure,imagewidth,imageheight,imagestructuresta
rt,imagestructureend)


for i=1+imagestructurestart:imageheight+imagestructureend
for j=1+imagestructurestart:imagewidth+imagestructureend
               w=1;
for m = imagestructurestart:imagestructureend
for n =imagestructurestart:imagestructureend
if (i-m>0)&(j-n>0)&(i-m<=imageheight)&(j-n<=imagewidth)
```

```
                        new(w)   = inputdata(i-m,j-n)+imagestructure(m,n);
                        w=w+1;
end
end
end
                news=median(new);
                data2(i-imagestructurestart,j-imagestructurestart)=news;
end
end
function
data2=Mero(inputdata,imagestructure,imagewidth,imageheight,imagestructurestar
t,imagestructureend)
for i=1-imagestructurestart:imageheight-imagestructureend
for j=1-imagestructurestart:imagewidth-imagestructureend
               w=1;
for m = imagestructurestart:imagestructureend
for n =imagestructurestart:imagestructureend
                      new(w)   = inputdata(i+m,j+n)-imagestructure(m,n);
                      w=w+1;
end
end
                news=median(new);
               data2(i+imagestructurestart,j+imagestructurestart)=news;
end
end
function [p]=newpp(p,v)
p=p+v;
if p>1.04
    p=1.04;
elseif p<0.15
    p=0.15;
else
    p=p;
end
function [v]=newv(t,v,p,pbest,gbest,iterator)
vmax=0.3*pi;
w=0.9-t*(0.8/iterator);
a1=pbest-p;
```

```
a2=gbest-p;
tempv=w.*v+1.4962*rand(1,1).*a1+1.4962*rand(1,1).*a2;
if tempv>vmax
    v=vmax;
elseif tempv<-vmax
    v=-vmax;
else
    v=tempv;
end
function Psnr = evaluate(data2,output)
[m0,n0]=size(output);
[m1,n1]=size(data2);
 m=min(m0,m1);
 n=min(n0,n1);
 x=0;
 z=m*n;
for h=1:m
for j=1:n
   x=x+(double(data2(h,j))-double(output(h,j)))^2;
end
end
mse=x/z;
maxium=255*255;
Psnr=10*log10(maxium/mse);
```

Appendix A3

```
performing a non maximum suppression to reduce the thich edges in the
%images.%%%%%%%%%%%%
function thin = Sup_thick(Image,fsize,lowThresh,highThresh)
[row,column] = size(Image);
thin = zeros(row,column);
clrimg = fspecial('average',[fsize,fsize]);
  ImSmooth = imfilter(Image,clrimg,'replicate');
fy = [3 4 4;4,4 3;4 4 4 ]; %the  step collection scheme used was used to find
the Vectro gradient of the image
fx = [3 4 4;3 4,5;4 4 4]; %the  roof collection scheme used was used to find
the  vector gradient of the image
ax = imfilter(ImSmooth,fx,'replicate');
ay = imfilter(ImSmooth,fy,'replicate');
mag = sqrt((ax.*ax) + (ay.*ay));
```

```
magmax = max(mag(:));
if magmax>0
    mag = mag / magmax;    % normalize
end
  Strongedge = [];
for dir = 1:4
    MaxI = Nms(dir,ax,ay,mag);
    Weakedge = MaxI(mag(MaxI) > lowThresh);
    thin(Weakedge)=1;
    Strongedge = [Strongedge; Weakedge(mag(Weakedge) > highThresh)];
end
if ~isempty(Strongedge) % result is all zeros if Strongedge is empty
    rstrong = rem(Strongedge-1, row)+1;
    cstrong = floor((Strongedge-1)/row)+1;
    thin = bwselect(thin, cstrong, rstrong, 8); % find the objects contain
pexel (cstrong(k),rstrong(k))
    thin = bwmorph(thin, 'thin', 1);  % Thin double (or triple) pixel wide
contours
end
end
function MaxI = Nms(direction,ix,iy,mag)
[m,n] = size(mag);
switch direction
case 1
  idx = find((iy<=0 & ix>-iy)   | (iy>=0 & ix<-iy));
case 2
  idx = find((ix>0 & -iy>=ix)   | (ix<0 & -iy<=ix));
case 3
  idx = find((ix<=0 & ix>iy) | (ix>=0 & ix<iy));
case 4
  idx = find((iy<0 & ix<=iy) | (iy>0 & ix>=iy));
end
% Exclude the exterior pixels
if ~isempty(idx)
  v = mod(idx,m);
  extIdx = (v==1 | v==0 | idx<=m | (idx>(n-1)*m));
  idx(extIdx) = [];
end
```

```
ixv = ix(idx);
iyv = iy(idx);
gradmag = mag(idx);
% Do the linear interpolations for the interior pixels
switch direction
case 1
  d = abs(iyv./ixv);
  gradmag1 = mag(idx+m).*(1-d) + mag(idx+m-1).*d;
  gradmag2 = mag(idx-m).*(1-d) + mag(idx-m+1).*d;
case 2
  d = abs(ixv./iyv);
  gradmag1 = mag(idx-1).*(1-d) + mag(idx+m-1).*d;
  gradmag2 = mag(idx+1).*(1-d) + mag(idx-m+1).*d;
case 3
  d = abs(ixv./iyv);
  gradmag1 = mag(idx-1).*(1-d) + mag(idx-m-1).*d;
  gradmag2 = mag(idx+1).*(1-d) + mag(idx+m+1).*d;
case 4
  d = abs(iyv./ixv);
  gradmag1 = mag(idx-m).*(1-d) + mag(idx-m-1).*d;
  gradmag2 = mag(idx+m).*(1-d) + mag(idx+m+1).*d;
end
MaxI = idx(gradmag>=gradmag1 & gradmag>=gradmag2);
end
```