IMPROVED EDGE DETECTION TECHNIQUE FOR DIGITAL IMAGES

Thesis submitted in partial fulfillment of the requirement for

The award of the degree of

Masters of Science

In

Mathematics and Computing

Submitted by

Parneet Kaur

Roll No.- 30703014

Under

the guidance of

Mr. Singara Singh

JULY 2009

School of Mathematics and Computer Applications

Thapar University

Patiala-147004 (PUNJAB)

INDIA

Improved Edge Detection Technique For Digital Images

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled "Improved Edge Detection technique for digital images" in partial fulfilment of the requirements for the award of degree of Master of Mathematics and Computing, School of Mathematics and Computer Applications, Thapar University, Patiala is an authentic record of my own work carried out under the supervision of Mr. Singara Singh.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Parneet Kaur

(Parneet Kaur)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Singara

(Mr. Singara Singh)
Lecturer
SMCA, Thapar University
Patiala

Contersigned by:

Dr S.S Bhatia 15.5.09
(Professor and Head)
SMCA, Thapar University,
Patiala

R K Sharma

Dr R.K Sharma 21/7/09
(Dean of Academic Affairs)
Thapar University
Patiala

ACKNOWLEDGEMENT

The completion of this thesis has involved a lot of people to whom I would like to express my sincere thanks and gratitude for their help.

For most, I would like to pass my appreciation and gratitude to my supervisor Mr. Singara Singh, Senior Lecturer, School of Mathematics and Computer Applications, Thapar University, Patiala for his constructive suggestions detailed corrections, support and encouragement in complishing this research work. Moreover, for mentoring me when I needed it the most. I am fortunate that I got an opportunity to work under his supervision.

I express my regards and gratitude to Dr S.S. Bhatia, Head of Department School of Mathematics and Computer Applications, Thapar University, Patiala for providing keen interest unfailing support, inspiration and necessary research facilities in the school.

I would like to thank my beloved parents and husband for their unconditional support and deep interest in me, without whom my project would have been a mere dream rather than a reality.

I would also thank all the academic and administrative staff of School of Mathematics and Computer Applications, Thapar University, Patiala.

Finally, I am thankful to all my friends who also contributed a lot in accomplishing this peace work.

(PARNEET KAUR)

Abstract

Sobel which is a popular edge detection method uses the derivative approximation to find edges. Therefore, it returns edges at those points where the gradient of the considered image is maximum. The edges determined by classic Sobel edge detector are thick. Then edge thinning technique is used to improve the results. The comparative experiment results show that the new edge detection technique for digital images is very effective. The results are also better than the classic Sobel edge detection method.

CONTENTS

Certificate ……i

Acknowledgment……..ii

Abstract……iii

Table of Contents……iv

List of Figures ……....v

Chapter 1 :Introduction…….....1

1.1 Introduction to the Edge Detection.……1

1.2 Factors Affecting the Selection of Edge Detectors……2

Chapter 2 :Different Types of Edge Detectors…….....3

2.1 Gradient Edge detection.……..3

2.2 Laplacian Edge Detection.……...7

2.3 Thresholding.……..10

2.3.1 Global Thresholding ……10

2.3.2 Local Thresholding.……...11

2.3.3 Adaptive Thresholding

2.4 Sobel Edge Detector.....16

2.5 Robert Edge Detector.....21

2.6 Canny Edge detector.....24

2.7 Prewitt Edge Detector.....28

Chapter 3: Problem Statement.....32

Chapter 4 :Proposed Solution

4.1 Proposed Algorithm.....33

4.2 Visual Comparisons of the Results.....36

Chapter 5: Conclusion and Future Work..... 43

Chapter 6 : References.....44

List Of Figures

Fig. 2.1: Original Image..... 5

Fig.2.2: Gradient of an Image..... 6

Fig.2.3: Original Lily flower Image..... 6

Fig.2.4: Gradient of Lily flower Image..... 7

Fig.2.5: Original shapes Image..... 8

Fig.2.6: Image after Laplacian mask..... 8

Fig.2.7: Original skelton Image..... 9

Fig.2.8: Image after Laplacian mask..... 9

Fig.2.9: Original Cane Image..... 18

Fig.2.10: Image after Sobel Algoriothm..... 18

Fig.2.11: Original Shapes Image..... 18

Fig.2.12: Image after Sobel Algorithm..... 18

Fig.2.13: Original Saturn Image..... 19

Fig.2.14: Image after Sobel Algorithm..... 19

Fig.2.15: Original Cman Image..... 19

Fig.2.16: Image after Sobel Algorithm..... 19

Fig.2.17: Original Tourist Image..... 20

Fig.2.18: Image after Sobel Algorithm..... 22

Fig.2.19: Original Cman Image..... 22

Fig.2.20: Image after Robert Algorithm..... 22

Fig.2.21: Original Blood Image..... 23

Fig.2.22: Image after Robert Algorithm..... 23

Fig.2.23: Original Cell Image..... 23

Fig.2.24: Image after Robert Algorithm..... 23

Fig.2.25: Original Image..... 23

Fig.2.26: Image after Robert Algorithm..... 23

Fig.2.27: Original Testpat Image..... 24

Fig.2.28: Image after Robert Algorithm..... 24

Fig.2.29: Original Rice Image..... 27

Fig.2.30: Image after Canny Algorithm..... 27

Fig.2.31: Lena Image..... 27

Fig.2.32: Image after Canny Algorithm..... 27
Fig.2.33: Original Image..... 27
Fig.2.34: Image after Canny Algorithm..... 28
Fig.2.35: Original Alumn Image..... 28
Fig.2.36: Image after Canny Algorithm..... 28
Fig.2.37: Original Moon Image..... 28
Fig.2.38: Image after Prewitt Algorithm..... 28
Fig.2.39: Original Blood Image..... 28
Fig.2.40: Image after Prewitt Algorithm..... 30
Fig.2.41: Original Cman Image..... 30
Fig.2.42: Image after Prewitt Algorithm..... 30
Fig.2.43: Rice Original Image..... 31
Fig.2.44: Image After Prewitt Algorithm..... 31
Fig.2.45: Bacteria Original Image..... 31
Fig.2.46: Image After Prewitt Algorithm..... 31
Fig.4.1: Original Rice Image..... 36
Fig.4.2: Output Image Sobel Edge Detection..... 36
Fig.4.3: Output Image of Proposed Algorithm..... 37
Fig.4.4: Blood Original Image..... 37
Fig.4.5: Output Image of Sobel Edge Detection..... 37
Fig.4.6: Output Image of Proposed Algorithm..... 37
Fig.4.7: Original Circles Image..... 38
Fig.4.8: Output Image Sobel Edge Detection..... 38
Fig.4.9: Output Image of Proposed Algorithm..... 38
Fig.4.10: Original Cman Image..... 39
Fig.4.11: Output Image Sobel Edge Detection..... 39
Fig.4.12: Output Image of Proposed Algorithm..... 40
Fig.4.13: Original Coins Image..... 41
Fig.4.14: Output Image Sobel Edge Detection..... 41
Fig.4.15: Output Image of Proposed Algorithm..... 42

Chapter 1

1.1 Introduction

A very important goal of computer image analysis and processing is to generate some particular images that are more suitable for people or machines to observe and identify ([8] [2]). Image edges are the most basic features of an image. The so-called image edge refers to the most prominent part of partial intensity changes in images. Edge detection is a process of detecting areas of abrupt changes or discontinuities in some visual property (light intensity, texture or colour). It is a critical pre-processing step towards high-level image understanding. Edges are essentially surface boundary discontinuities, thus they hold important feature information about objects in an image (e.g. size, shape and location). Edges characterize boundaries and are therefore a problem of fundamental importance in image processing. Edges in images are areas with strong intensity contrasts – a jump in intensity from one pixel to the next. Edge detecting an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image. Important features can be extracted from the edges of an image (e.g., corners, lines, curves). There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories, Gradient and Laplacian. However, there are some problems with these methods such as influences of noises, false edges, missing edges, and missing corners and junction.

Edge detection is an important task in image processing. It is a main tool in pattern recognition, image segmentation, and scene analysis. An edge detector is basically a high pass filter that can be applied to extract the edge points in an image.

1.2 Factors affecting the selection of Edge Detectors

Edge orientation: The geometry of the operator determines a characteristic direction in which it is most sensitive to edges. Operators can be optimized to look for horizontal, vertical, or diagonal edges.

Noise environment: Edge detection is difficult in noisy images, since both the noise and the edges contain high-frequency content. Attempts to reduce the noise result in blurred and distorted edges. Operators used on noisy images are typically larger in scope, so they can average enough data to discount localized noisy pixels. This results in less accurate localization of the detected edges.

Edge structure: Not all edges involve a step change in intensity. Effects such as refraction or poor focus can result in objects with boundaries defined by a gradual change in intensity. The operator needs to be chosen to be responsive to such a gradual change in those cases. Newer wavelet-based techniques actually characterize the nature of the transition for each edge in order to distinguish, for examp

Chapter-2

Different types of edge detectors

There are many edge detection algorithms. The key of these edge detection algorithms is the choice of threshold; threshold directly determines the success of edge detection. How to automatically get the best threshold of edge has been one of the difficulties of edge detection? If the selected threshold is too low, it will not only generate false edges, but edges are very thick; these edges will need to be refined again and the locations of the reprocessed edges are often not precise enough. If the threshold is too high, then many of the edges may not be detected or the detected edges are too segmented.

A variety of Edge Detectors are available for detecting the edges in digital images. However, each detector has its own advantages and disadvantages. The basic idea behind edge detection is to find places in an image where the intensity changes rapidly. Based on this idea, an edge detector may either be based on the technique of locating the places where the first derivative of the intensity is greater in magnitude than a specified threshold or it may be based on the criterion to find places where the second derivative of the intensity has a zero crossing.

The popular edge detection operators are Roberts, Sobel, Prewitt and Canny operators. In certain situations where the edges are highly directional, some edge detector works especially well because their patterns fit the edges better.

The edge detection techniques grouped into two categories: Gradient edge detection and Laplacian edge detection. These are discussed in detail in the next subsections.

2.1 Gradient Edge detection

Major generic approaches to edge detection are model fitting and differentiation. Model fitting involves matching the actual image data against ideal edge model. If the fit is sufficiently close, an edge is assumed to exist. The differentiation approach has been the classical and most often used edge detection method. It is computationally less intensive than the model fitting methods. Edges are most frequently defined as areas where abrupt changes in light intensity occur. Hence mathematically they will manifest in the derivatives [1] of the image function. Differential method uses approximation of spatial gradient at each pixel location. Denote f (x, y) to be the function that maps gray scale value at a particular pixel a_0 to its Cartesian co-ordinates. Let G(x, y) be the rate of change in gray scale value at pixel a_0. G(x, y) can be computed in terms of the derivatives along x and y directions, $G_x(x, y)$ and $G_y(x, y)$ as follows:

$$G(x, y) = \left\{ \left[G_x(x, y) \right]^2 + \left[G_y(x, y) \right]^2 \right\}^{\frac{1}{2}}$$

This quantity gives the maximum rate of increase of f(x, y) per unit distance in the direction of G(x, y).

The direction of the gradient vector also is an important quantity. Let $\alpha(x, y)$ represent the direction angle of the vector G(x,y) at (x,y).

$$\alpha(x, y) = \tan^{-1} \frac{G_y(x, y)}{G_x(x, y)}$$

Computation of the gradient of an image is based on obtaining the partial derivatives $G_x(x,y)$ *and* $G_y(x,y)$ at every pixel location. Let the 3X3 kernels are:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Based on 3X3 convolution matrices, the result of images is shown below:

Fig. 2.1 Original Colouful Image

Fig 2.2 Gradient of an image

Fig 2.3 Original Lily flower image

Fig. 2.4 Gradient of lily flower image

2.2 Laplacian Edge Detection

Gradient operation is an effective detector for sharp edges where the pixel gray levels change over space very rapidly. But when the gray levels change slowly from dark to bright (red in the figure below), the gradient operation will produce a very wide edge (green in the figure). It is helpful in this case to consider using the Laplace operation. The Laplacian method searches for zero crossings [12] in the second derivative of the image to find edges[1]. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location.

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial^2 f}{\partial x^2} = f(i, j+1) - 2f(i, j) + f(i, j-1)$$

$$\frac{\partial^2 f}{\partial y^2} = f(i+1, j) - 2f(i, j) + f(i-1, j)$$

$$\nabla^2 f = -4f(i,j) + f(i,j+1) + f(i,j-1) + f(i+1,j) + f(i-1,j)$$

The Laplacian can be implemented using the mask shown below:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

The Laplacian is combined with smoothing as a precursor to finding edges via zero-crossings. Consider the function

$$h(r) = -e^{-\frac{r^2}{2\sigma^2}}$$

where $r^2 = x^2 + y^2$ and σ is the standard deviation. Convolving this function with an image blurs the image, with the degree of blurring being determined by the value of σ.The laplacian of h(second derivative of h with respect to r)is

$$\nabla^2 h(r) = -\left[\frac{r^2 - \sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}}$$

This function is commonly referred to as the Laplacian of a Gaussian (LoG) .The co-efficients of LoG mask must sum to zero. A mask is useful only for images that are essentially noise free.

Some image results of above Laplacian mask are shown in the following Images:

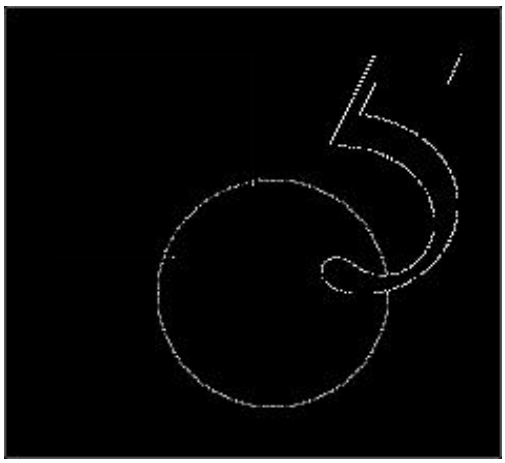


Fig. 2.5 Original shapes image Fig. 2.6 Image after Laplacian mask

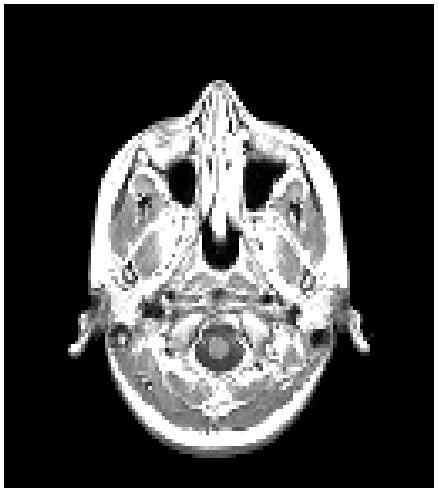


Fig. 2.7 Original skelton image

Fig 2.8 Image after Laplacian mask

Thus, we see that the purpose of the Gaussian formulation is to smooth the image, and the purpose of the laplacian operator is to provide an image with zero-crossings used to establish the location of edges. Smoothing the image reduces the effect of noise caused by the second derivative of the Laplacian.Comparing Figs.(2.2,2.4,2.5) reveals several important differences.First,we note that the edges in the zero-crossing image are thinner than the gradient edges. This is the main characterstic of zero-crossings that make this approach attractive. On the other hand, major drawback is the computation of zero-crossings. For this reason, edge-finding techniques based on various implementation of the gradient are used more frequently than zero-crossings in the implementation of segmentation logarithms.

2.3 Thresholding

How to quickly and accurately extract the edge information of the images always is a hot research topic. The key of classical edge detection algorithm is the choice of threshold; threshold directly determines the success of edge detection. How to automatically get the best threshold of edge has been one of the difficulties of edge detection. If the selected threshold is too low, it will not only generate false edges, but edges are very thick; these edges will need to be refined again and the locations of the reprocessed edges are often not precise enough. If the threshold is too high, then many of the edges may not be detected or the detected edges are too segmented. In this section, we discuss ways of choosing threshold value automatically and method for varying threshold according to the properties of local image neighbourhoods.

Let f(x, y) composed of light objects on a dark background in such a way that object and background pixels has intensity levels grouped into two modes. One way to extract the objects from the background is to select a threshold T that separate these modes. Then any point (x, y) for which $f(x, y) \geq T$ is called an object point otherwise point is called background point. The thresholded image g(x, y) is defined as

$$g(x, y) = \begin{cases} 1 & if \quad f(x, y) \geq T \\ 0 & if \quad f(x, y) < T \end{cases}$$

Pixels labelled 1 correspond to objects, whereas pixels labelled 0 correspond to the background. When T is a constant, this approach is called global thresholding.

2.3.1 Global Thresholding

Another method of choosing threshold T is by trial, picking different thresholds until one is found that produces good results. This is

particularly effective method that allows the user to change the threshold value ([1]-[2]). The type of global thresholding can be successful in highly controlled enviorments.One of the areas in which this often possible, where control of the illumination usually is feasible.

For choosing a threshold automatically, Gonzalez and Woods [1] describe the following iterative procedure:

I. Select an initial estimate for T (suggested initial estimate is the mid-point between the minimum and maximum intensity values in the image).

II. Segment the image using T. This will produce two groups of pixels G_1 with intensity values $\geq T$ and G_2 with intensity values <T.

III. Compute the average intensity values μ_1 *and* μ_2 for the pixels in regions G_1 *and* G_2.

IV. Compute a new threshold value:

$$T=\frac{1}{2}(\mu_1 + \mu_2)$$

Repeat steps 2 through 4 until the difference in T in successive iterations is smaller than a predefined parameter T_0.The parameter T_0 is used to stop the algorithm after changes become small in terms of this parameter.

2.3.2 Local Thresholding

Thresholding may be viewed as an operation that involves test against a function T of the form

$$T=T[x, y, p(x, y), f(x, y)]$$

where f(x , y) is the gray level of point (x, y) and p(x, y) denotes some local properties of this point-for example, the average gray level of a neighbourhood centred on (x , y).

When T depends only on f(x, y) the threshold is called global. If t depends on f(x, y) and p(x, y) ,the threshold is called local.

It is evident that the chances of good threshold are enhanced only if the pixels on or near the edge between object and the background were used. In addition the probability of pixels lie on an object would be approximately equal to the probability that it lies on the background. The Laplacian can yield information regarding whether a pixel lies on the dark or light side of an image.

The gradient ∇f at any point (x, y) in an image is given by $\nabla f = \sqrt{G_x^{\,2} + G_y^{\,2}}$ and the Laplacian $\nabla^2 f$ is given by $\nabla^2 f = -4f(i,j) + f(i,j+1) + f(i,j-1) + f(i+1,j) + f(i-1,j)$.

These two quantities may be used to form three –level image as follows:

$$s(x,y) = \begin{cases} 0 & \text{if } \nabla f < T \\ + & \text{if } \nabla f \geq T \text{ and } \nabla^2 f \geq 0 \\ - & \text{if } \nabla f \geq T \text{ and } \nabla^2 f < 0 \end{cases} \qquad (2.1)$$

where the symbols 0,+ and – represents three distinct gray levels,T is threshold, and gradient and Laplacian are calculated at every point (x, y).For a dark object on a light background and the use of (equation 2.1) produces an image s(x, y) in which ;(1) all pixels that are not on an edge(as determined by $\nabla f < T$)are labelled 0;(2) all pixels on the dark side of an edge are labelled +;and (3)all pixels on the light side of an edge are labelled -. The symbols + and – in equation 2.1 are reserved for a light object on a dark background.

The information obtained with this procedure can be used to generate a segmented, binary image in which 1's correspond to objects and 0's correspond to the background. The transition from a light background to a dark object must be characterised by the occurrence of a-

followed by a+ in s(x , y).The interior of the object is composed of pixels that are labelled either 0 or +.Thus a horizontal or vertical line containing a section of an object has the following structure:

(…) (-,+) (0 or +) (+,-) (…)

Where (…)represents any combination of +,- and 0.The innermost parenthesis contain object points and are labelled 1.All other pixels along the same scan line are labelled 0,with the exception of any other sequence of (0 or +) bounded by (-.+) and (+,-).

2.3.3 Adaptive Thresholding

Imaging factors such as uneven illumination can transform a perfectly segmentable histogram into a histogram that cannot be portioned effectively by a single global threshold. An approach for handling such a situation is to divide original image into subimages and then utilise a different threshold to segment each subimage. Since the threshold for each pixel depends on the location of pixel in terms of sub images, this type of thresholding is adaptive thresholding.

Suppose that an image contains only two principal gray-level regions. Let z denote gray-level values. Their histogram may be considered as estimate of their probability density function (PDF) p(z).This overall density function is the sum of two densities, one for light and the other for the dark regions in the image.

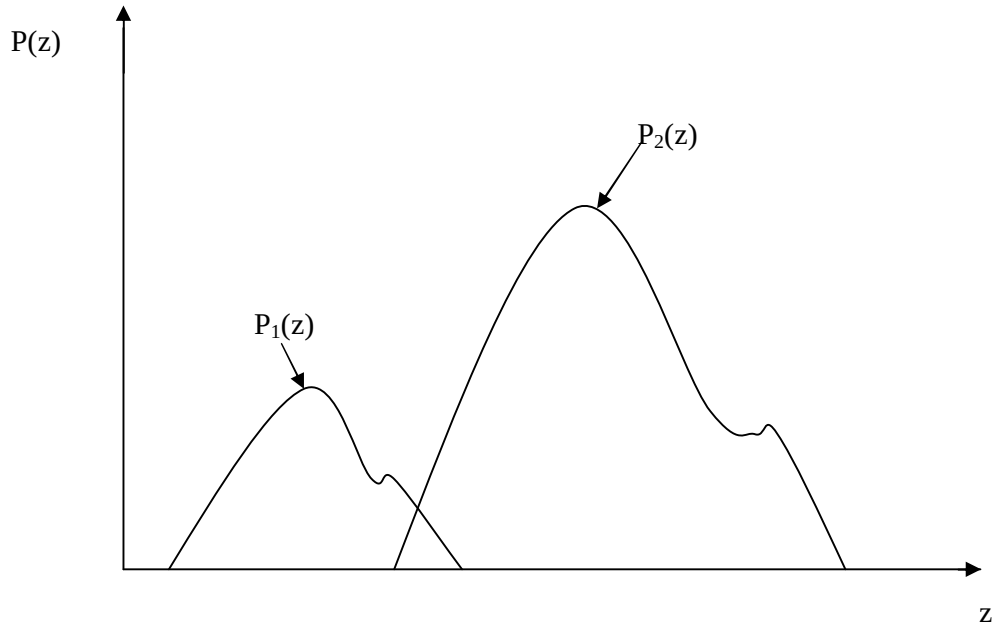


Figure 3.

Figure 3 shows two probability density functions. Assume that the larger of the two PDFs corresponds to the background levels while the smaller describes the gray levels of objects in the image. The sum probability density function describing the overall gray-level variation in the image is

$$p(z) = P_1\ p_1(z) + P_2\ \ p_2(z)$$

Here, P_1 *and* P_2 are the probabilities of occurrence of two classes of pixels; P_1 is the probability that a random pixel with value z is an object pixel. Similarly, P_2 is the probability that the pixel is a background pixel. We are assuming that any given pixel belongs either to an object or to the background, so that

$$P_1 + P_2 = 1$$

An image is segmented by classifying as background all pixels with gray levels greater than a threshold T. All other pixels are called object pixels. Our main objective is to select the value T that minimises the average

error making the decision that a given pixel belongs to an object or to the background.

The probability of a random variable having a value in the interval [a, b] is the integral of its probability density function from a to b, which is the area of the PDF curve between these two limits. Thus, the probability of background point as an object point is

$$E_1(T) = \int_{-\infty}^{T} p_2(z)dz$$

This is the area under the curve of $p_2(z)$ to the left of the threshold. Similarly, the probability of an object point as background is

$$E_2(T) = \int_{T}^{\infty} p_1(z)dz$$

which is the area under the curve of $p_1(z)$ to the right of T. Then overall probability of error is

$$E(T) = P_2 E_1(T) + P_1 E_2(T)$$

In the extreme case in which background points are known never to occur. In this case $P_2 = 0.$ The contribution to the overall error (E) of classifying a background point as an object point (E_1) should be zeroed out because background points are known never to occur. If background and object points are equally likely to occur, then the weights are $P_1 = P_2 = 0.5$. If $P_1 = P_2$, the optimal threshold is the average of the means.

Obtaining an analytical expression for T requires that we know the equations for the two PDF'S. Estimating these densities in practice is not always feasible, and an approach used often is to employ densities whose parameters are simple to obtain. One of the principal densities used is the Gaussian density, which is characterized by two parameters; the mean and variance.

$$p(z) = \frac{P_1}{\sqrt{2\prod}\,\sigma_1} e^{-\frac{(z-\mu_1)^2}{2\sigma_1^2}} + \frac{P_2}{\sqrt{2\prod}\,\sigma_2} e^{-\frac{(z-\mu_2)^2}{2\sigma\sigma_2^2}}$$

Where, μ_1 *and* σ_1^2 are the mean and variance of Gaussian density function of one class of pixels and μ_2 *and* σ_2^2 are the mean and variance of the other class.

Using this equation in general solution that is $P_1\ p_1(T) = P_2\ p_2(T)$ results in the following solution for the threshold T:

$$AT^2 + BT + C = 0$$

where,

$$A = \sigma_1^2 - \sigma_2^2$$
$$B = 2(\mu_1 \sigma_2^2 - \mu_2 \sigma_1^2)$$
$$C = \sigma_1^2 \mu_2^2 - \sigma_2^2 \mu_1^2 + 2\sigma_1^2 \sigma_2^2 \ln(\sigma_2 P_1 / \sigma_1 P_2)$$

Since a quadratic equation has two possible solutions, two threshold values may be required to obtain optimal solution.

If the variance is equal, i.e. $\sigma^2 = \sigma_1^2 = \sigma_2^2$, then a single threshold is sufficient:

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} \ln(\frac{P_2}{P_1})$$

If $P_1 = P_2$, then optimal threshold is average of the means. The same is true if $\sigma = 0$.

2.4 SOBEL EDGE DETECTOR

The Sobel operator [11] is widely used in image processing, particularly within edge detection algorithms. Sobel edge detector is one of the classic [4] edge detector. Technically, it is a discrete differentiation operator,

computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is the corresponding gradient vector. Get the gradient of the image by using the Sobel's kernels given below.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

and

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels are separately convolved with the input image, to produce separate measurements of the gradient component in each orientation (call these G_x and G_y). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

Typically, an approximate magnitude is computed using:

$$|G| = |G_x| + |G_y|$$

These approximations still behave as derivatives, that is they are zeroes in areas of constant intensity and their values are proportional to the degrees of intensity change in areas whose pixel values are variables. It is common practice to refer to the magnitude of the gradient which is much faster to compute.

Then, we say that a pixel at location (x, y) is an edge pixel if G>=T at that location, where T is a specific threshold.

The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by:

$$\theta = \tan^{-1}(\frac{G_y}{G_x})$$

In order to test the effectiveness of the algorithm, we implemented in matlab:

Fig. 2.9 Original cane image

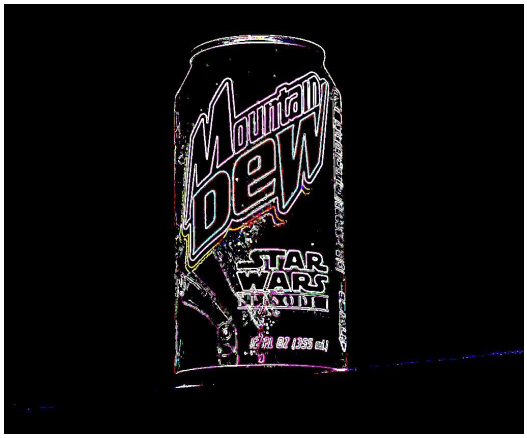


Fig 2.10 Image after Sobel algorithm

Fig. 2.11 Original shapes image

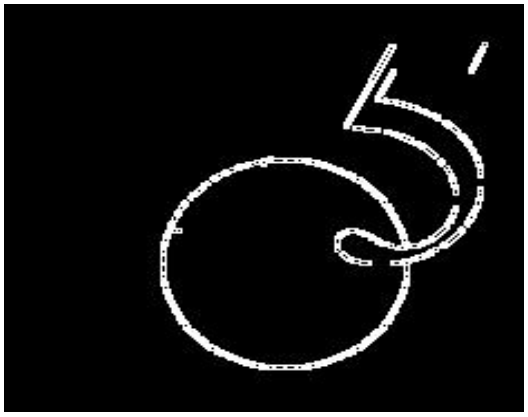


Fig. 2.12 Image after Sobel algorithm

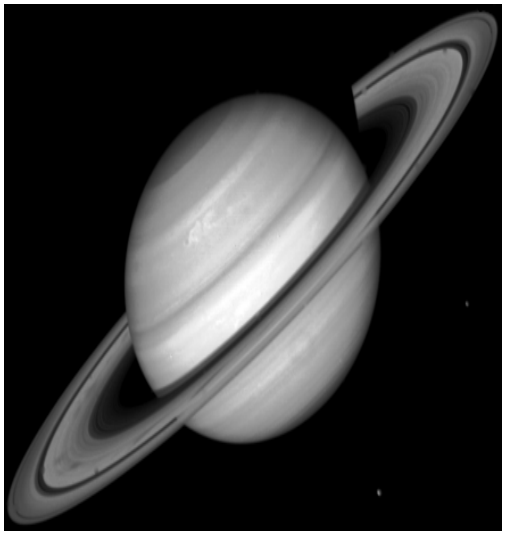


Fig 2.13 Original Saturn image

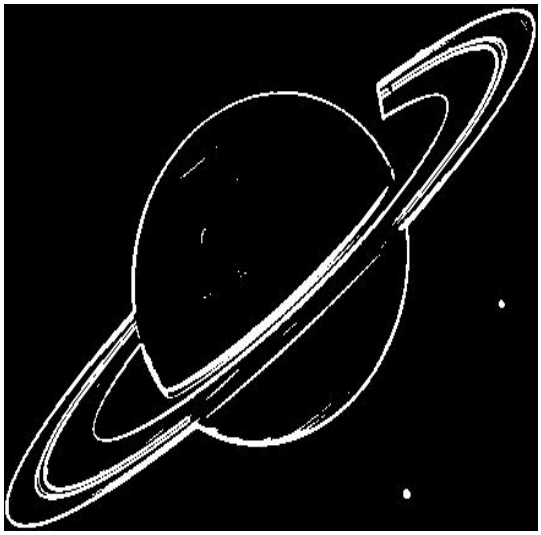


Fig 2.14 Image after Sobel algorithm

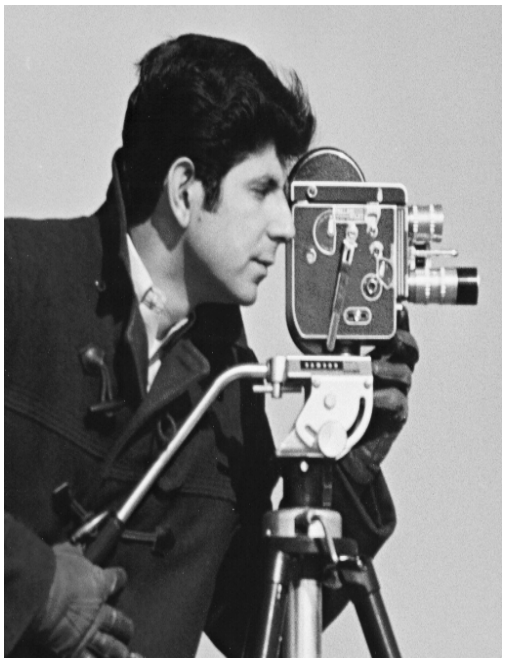


Fig. 2.15 Original cman image

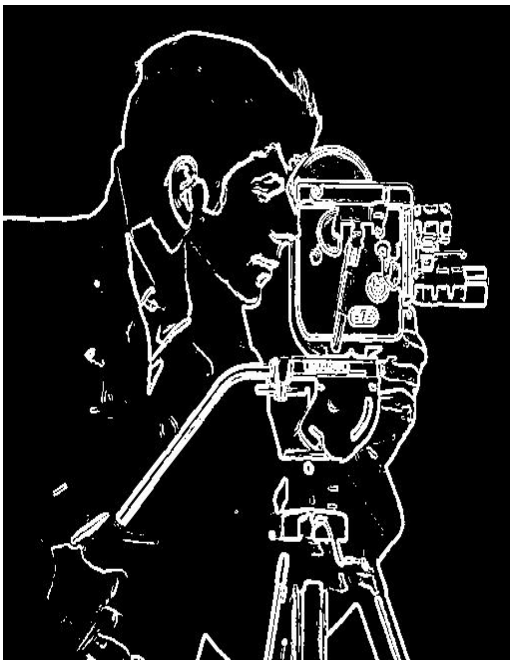


Fig. 2.16 Image after Sobel algorithm

Fig. 2.17 Original tourist image

Fig. 2.18 Image after Sobel edge detection algorithm

2.5 **ROBERT EDGE DETCTOR**

The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image. The Roberts edge detector is one of the oldest edge detectors in digital image processing and it is also the simplest. This detector is used considerably less than the others due to its limited functionality(e.g., it is not symmetric and cannot be generalized to detect edges that are multiple of 45^0).Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point.

The operator consists of a pair of 2×2 convolution kernels as shown below. One kernel is simply the other rotated by 90°. This is very similar to the Sobel operator.

$$G_x = \frac{\partial f}{\partial x} = f(i, j) - f(i+1, j+1)$$

$$G_y = \frac{\partial f}{\partial y} = f(i+1, j) - f(i, j+1)$$

This approximation can be implemented by the following masks:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad \textit{and} \quad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

These kernels are designed to respond maximally to edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be convolved separately with the input image, to produce separate measurements of the gradient component in each orientation (call these Gx and Gy). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{G_x^{\,2} + G_y^{\,2}}$$

although typically, an approximate magnitude is computed using:

$$|G| = |G_x| + |G_y|$$

These approximations still behave as derivatives, that is they are zeroes in areas of constant intensity and their values are proportional to the degrees of intensity change in areas whose pixel values are variables. It is common practice to refer to the magnitude of the gradient which is much faster to compute.

Then, we say that a pixel at location (x, y) is an edge pixel if G>=T at that location, where T is a specific threshold.

The angle of orientation of the edge giving rise to the spatial gradient (relative to the pixel grid orientation) is given by:

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right) - \frac{3\pi}{4}$$

Some image results of Robert edge detector are given in the following images:

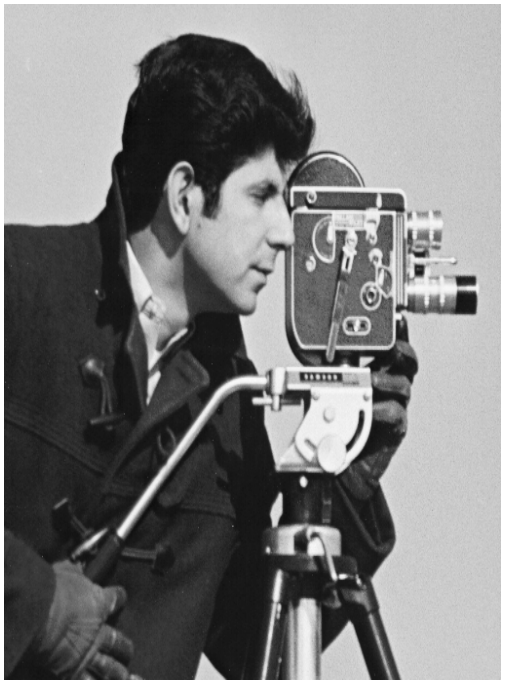


Fig. 2.19 Original cman image

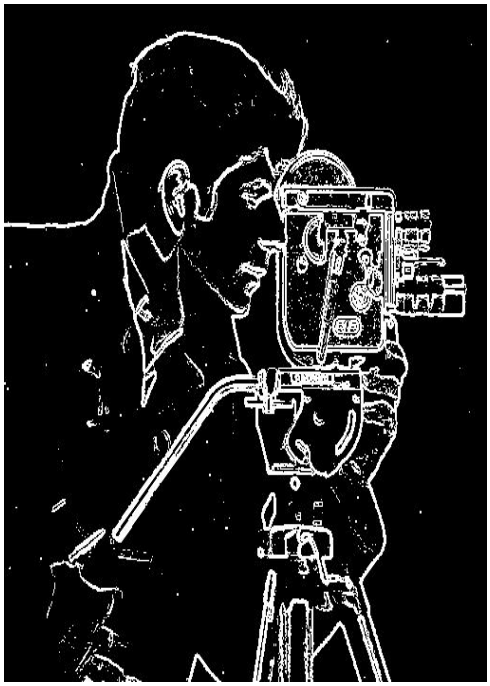


Fig. 2.20 Image after Robert algorithm

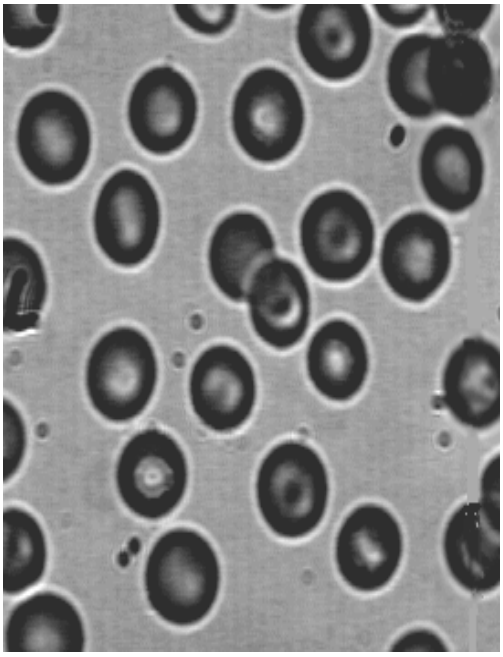


Fig.2.21 Original blood Image

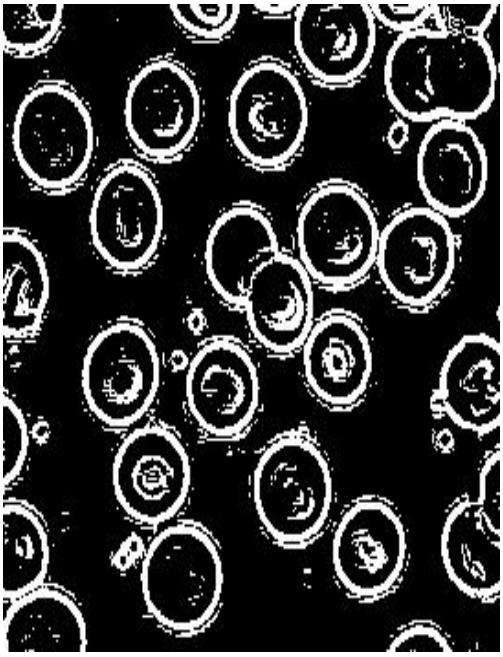


Fig 2.22 Image after Robert algorithm

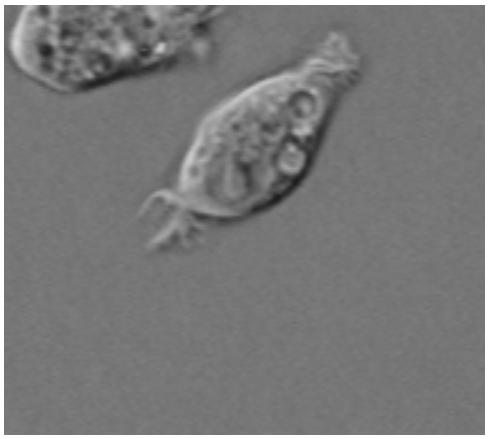


Fig. 2.23 Original cells image

Fig 2.24 Image after Robert algorithm

Fig.2.25 Original image

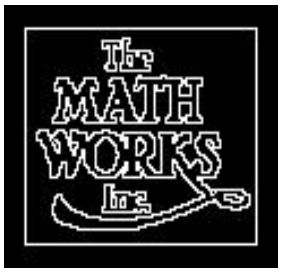
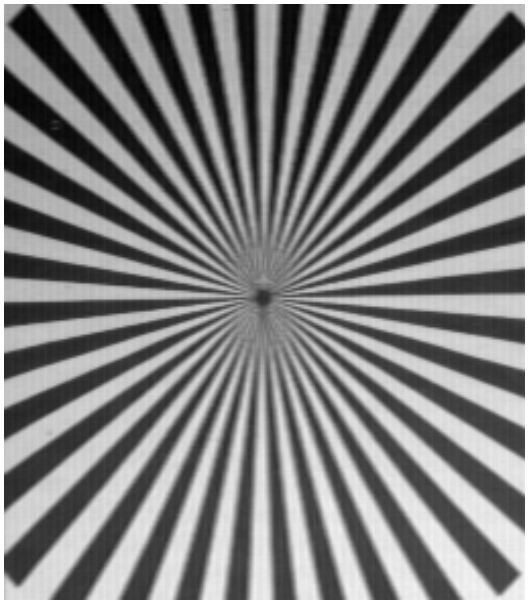


Fig.2.26 Image after Robert algorithm



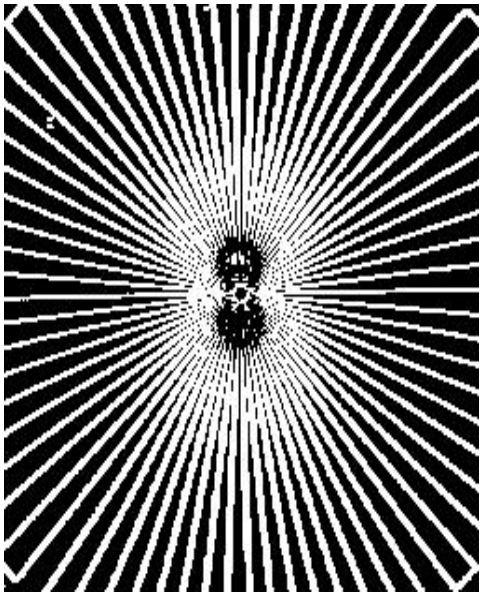


Fig. 2.27 Original testpat image Fig2.28 Image after Robert algorithm

2.6 **Canny's Edge Detector**:

The Canny edge detector [7] is regarded as one of the best edge detectors currently in use. The basic algorithm can be outlined as follows:

I. Blur the image slightly. This removes the effects of random black or white pixels (i.e. noise pixels) that can adversely affect the following stages. Blurring is typically accomplished using a Gaussian Blur.

$$\frac{1}{256}\begin{bmatrix} 2 & 4 & 4 & 4 & 2 \\ 4 & 8 & 16 & 8 & 4 \\ 4 & 16 & 32 & 16 & 4 \\ 4 & 8 & 16 & 8 & 4 \\ 2 & 4 & 4 & 4 & 2 \end{bmatrix}$$

Gaussian Mask with Standard Deviation of 1.36

- The size and the standard deviation are two important criteria that need to be taken into consideration when creating the Gaussian mask. Larger sizes of the Gaussian mask will mean that the edge detection will be less susceptible to noise; however, larger mask also leads to more calculations and therefore a slower performance. The standard deviation should be chosen in such a way that the Gaussian mask provides an accurate representation of a Gaussian curve. The mask only needs to represent the first three standard deviations of the Gaussian curve. Convolution with numerical values outside of the three standard deviation range has little effect on the image. If the standard deviation is too small, the three standard deviation range will not be accurately represented; if too big, extra calculations with little contribution will be performed.

II. Compute the x and y derivatives of the image. This is basically an edge detection step. Similar to the Sobel edge detector the x and y derivatives are simply calculated by subtracting the values of the two surrounding neighbouring pixels depending on which axis is being computed (left and right for x derivative and top and bottom for y derivative).

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

The magnitude of the gradient is then approximated using the formula:

$$|G| = |G_x| + |G_y|$$

III. When the gradient in the x direction is equal to zero, the edge direction has to be equal to 90 degrees or 0 degrees, depending on what the value of the gradient in the y-direction is equal to. If G_y has a value of zero, the edge direction will equal 0 degrees. Otherwise the edge direction will equal 90 degrees. The formula for finding the edge direction is :

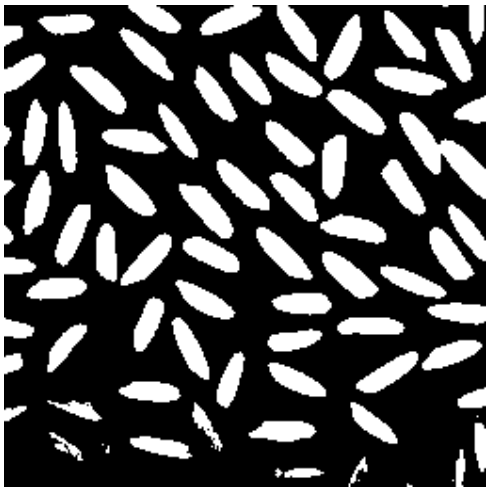
$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

IV. Edge thinning is performed by computing gradient magnitude in four possible directions when describing the surrounding pixels-0 degrees(in the horizontal direction), 45^0 (along the positive diagonal), 90^0 (in the vertical direction), or 135^0 (along the negative diagonal). So now the edge orientation has to be resolved into one of these four directions depending on which direction it is closest to (e.g. if the orientation angle is found to be 3 degrees, make it zero degrees). Therefore, any edge direction falling within the range (0 to 22.5 & 157.5 to 180 degrees) is set to 0 degrees. Any edge direction falling in the range (22.5 to 67.5 degrees) is set to 45 degrees. Any edge direction falling in the range (67.5 to 112.5 degrees) is set to 90 degrees. And finally, any edge direction falling within the range (112.5 to 157.5 degrees) is set to 135 degrees.

V. Once the edges have been thinned the last step is to threshold the detected edges. Edge magnitudes above the upper threshold are preserved. Edges below the upper threshold but above the lower threshold are preserved only if they connect to edges that are above the upper threshold. This process is know as hysteresis and allows

edges to grow larger than they would by using a single threshold without introducing more noise into the resulting edge image.

Some image results of Canny edge detection are shown in the following figures:



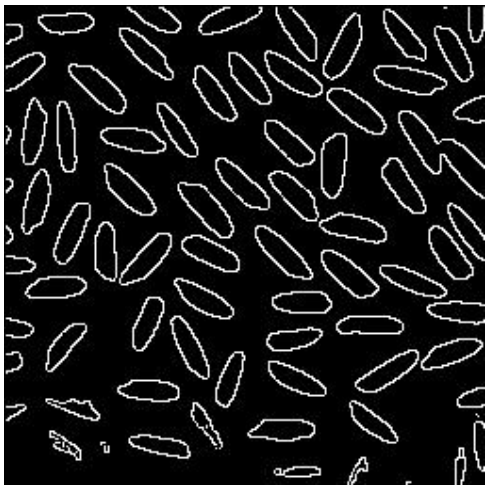


Fig 2.29 Original rice image Fig 2.30Image after Canny algorithm

Fig.2.31 Lena image

Fig. 2.32Image after Canny algorithm

Fig 2.33 Original image

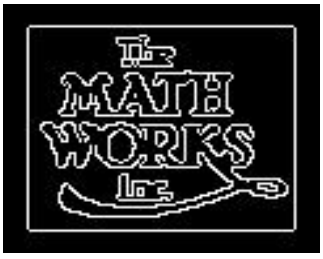


Fig 2.34Image after Canny algorithm

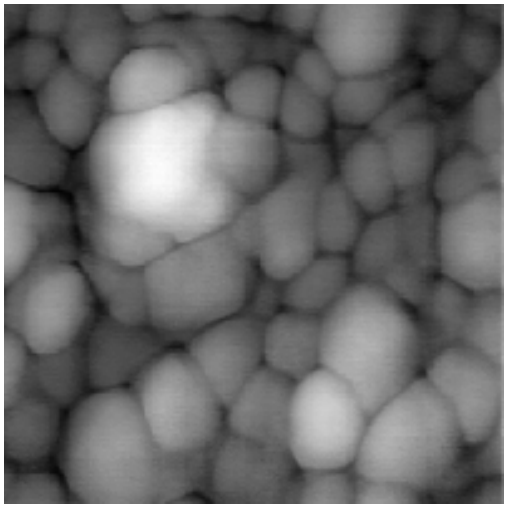


Fig. 2.35 Original alumn image

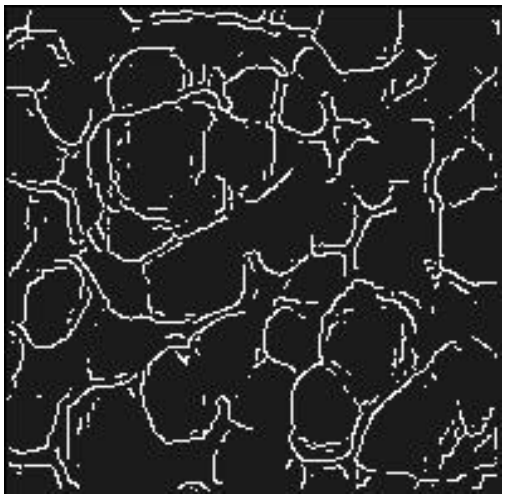


Fig 2.36Image after Canny algorithm

Fig 2.37 Original moon image

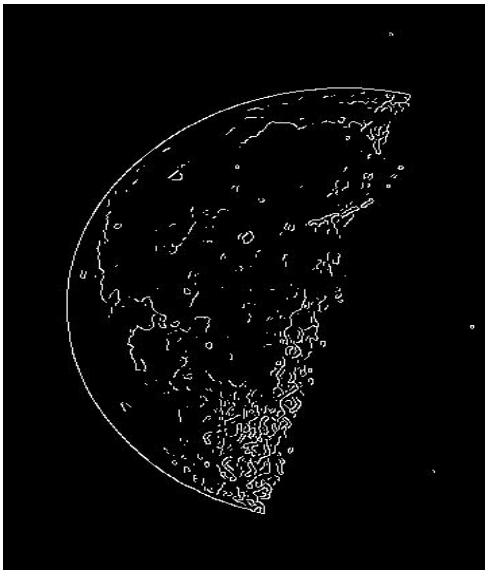


Fig 2.38 Image after Canny algorithm

2.7Prewitt edge detector:

The Prewitt operator [10] is widely used in image processing, particularly within edge detection algorithms. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of

the image intensity function. At each point in the image, the result of the Prewitt operator is the corresponding gradient vector.

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ -1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be separately convolved with the input image, to produce separate measurements of the gradient component in each orientation (call these G_x and G_y). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{G_x^{\,2} + G_y^{\,2}}$$

Typically, an approximate magnitude is computed using:

$$|G| = |G_x| + |G_y|$$

These approximations still behave as derivatives, that is they are zeroes in areas of constant intensity and their values are proportional to the degrees of intensity change in areas whose pixel values are variables. It is common practice to refer to the magnitude of the gradient which is much faster to compute.

Then, we say that a pixel at location (x, y) is an edge pixel if G>=T at that location, where T is a specific threshold.

The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by:

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Some image results of Prewitt edge detector are shown in following figures:

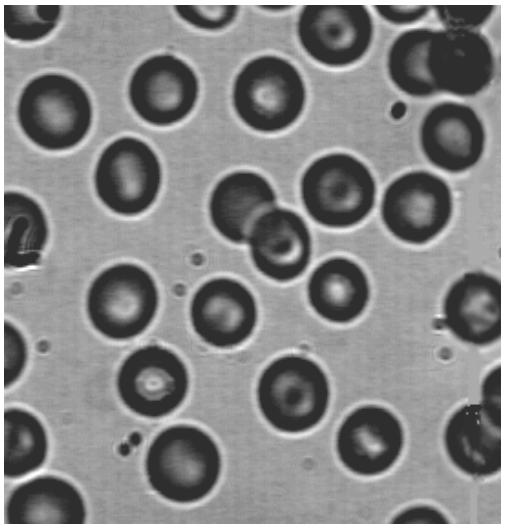


Fig 2.39 Original blood image

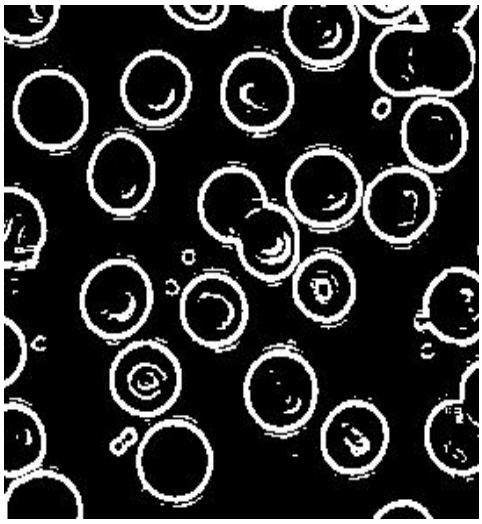


Fig 2.40 Image after Prewitt algorithm

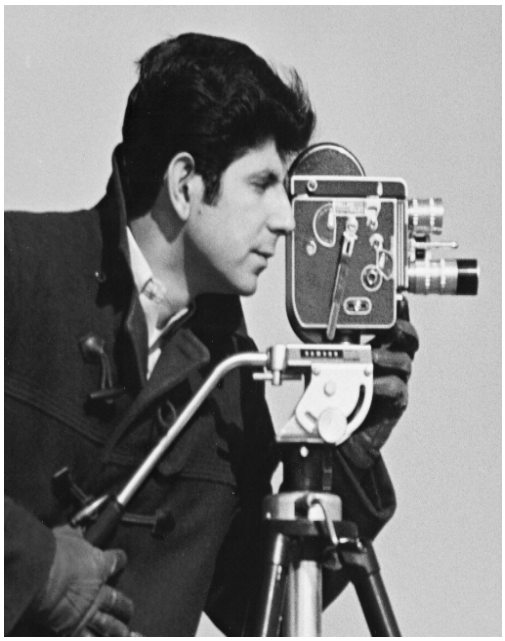


Fig 2.41 Original cman Image

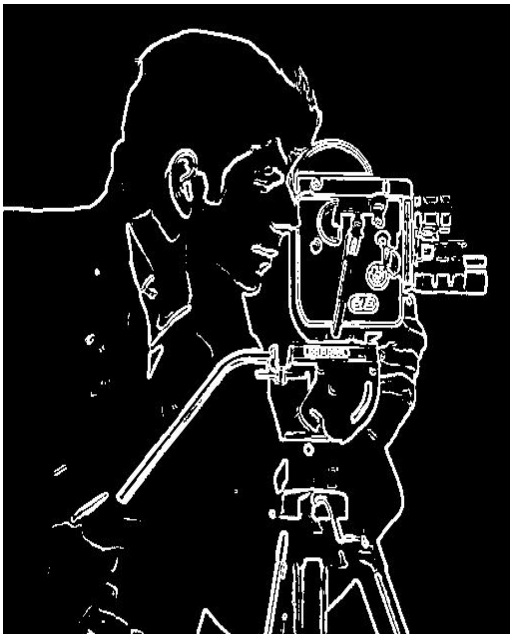


Fig 2.42 Image after Prewitt algorithm

Fig 2.43 Rice original image

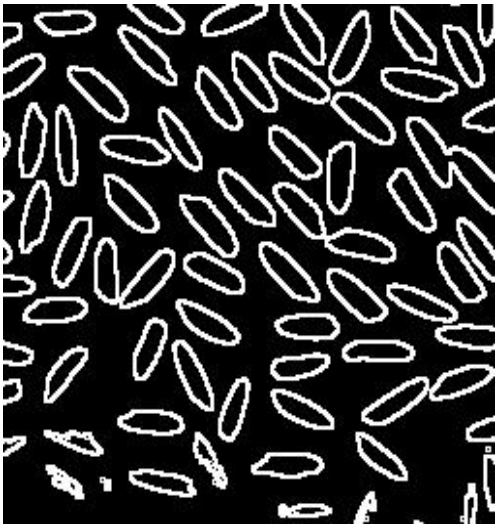


Fig 2.44 Image after Prewitt algorithm

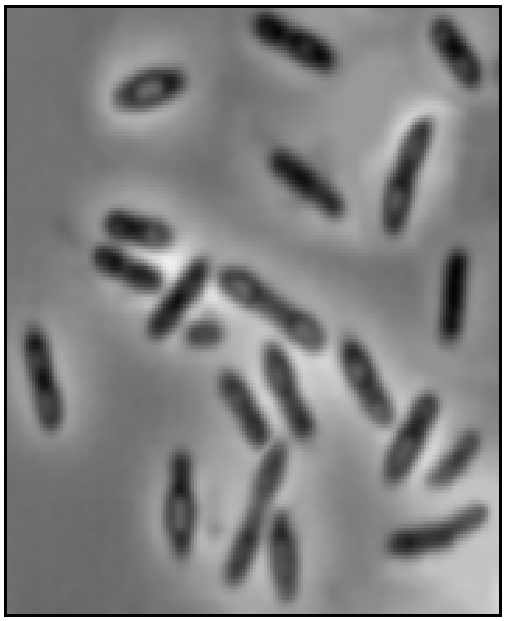


Fig 2.45 Bacteria original image

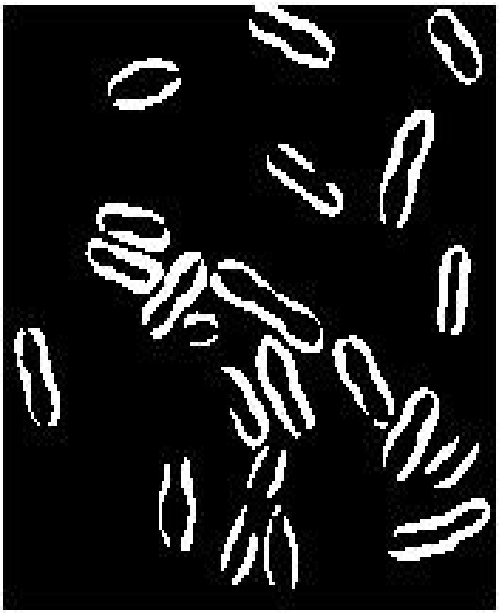


Fig 2.46 Image after Prewitt algorithm

Chapter-3

Problem Statement

Effective edge detection is required for many important areas such as machine vision and automated interpretation systems and is often used as the front end processing stage in object recognition and interpretation systems. An edge detector is defined as a mathematical operator of small spatial extent that responds in some way to these discontinuities usually classifying every image pixel as either belonging to an edge or not. Traditional image edge detectors commonly extract edges by adopting specific templates, or in combination with smoothing functions. However, traditional edge filtering methods often result in some drawbacks like broken edges, thick edges and false edges. Therefore many methods have been proposed to link these broken edges in order to improve edge detection. In this thesis, we proposed a method to improve the thick edges.

In our proposed algorithm, Sobel's edge detector filters are applied in the horizontal and vertical directions and Otsu's threshold is used to determine edge or non-edge pixels, and then on the output image, a thinning process is applied to smooth the thick edges. The visual comparison of the results of the proposed algorithm with the results of the already existing algorithms shows the effectiveness of the proposed algorithm. The proposed algorithm is explained in the next section.

Chapter-4

4.1 **Proposed Algorithm**

The proposed algorithm mainly consists of three steps explained below:

1. Get the gradient of the image by using the Sobel's kernels given below.

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

and

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

These kernels are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels are convolved separately with the input image, to produce separate measurements of the gradient component in each orientation (call these G_x and G_y). These can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

Typically, an approximate magnitude is computed using:

$$|G| = |G_x| + |G_y|$$

These approximations still behave as derivatives, that is they are zeroes in areas of constant intensity and their values are proportional to the degrees of intensity change in areas whose pixel values are variables.

It is common practice to refer to the magnitude of the gradient which is much faster to compute.

Then, we say that a pixel at location (x, y) is an edge pixel if G>=T at that location, where T is a specific threshold.

The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by:

$$\theta = \tan^{-1}(\frac{G_y}{G_x})$$

2 Resolve the best partition threshold of the gradient image by Otsu's method[5].

2.1 Otsu thresholding:

There are many threshold selection methods. Depending on the scope of applications, the threshold methods can be divided into the 1) global threshold method, 2) local threshold and 3) dynamic threshold method. New proposed edge detection algorithm belongs to the Global Otsu threshold method ([12]-[14]).

Suppose f (x, y) is the objective image that we want to segment, its gray range is {0, 1 ,….., L-1}. The image pixels will be divided into two categories by threshold t: C0 = {0, 1,…, t}, C1 = {t +1, t +2, …, L-1}. C0 and C1 respectively represent the target and background. The Classes square error between C0 and C1 as follows:

$$\sigma(t)^2 = \omega_0(t) * \varpi_1(t) * (\mu_0(t) - \mu_1(t))^2$$

Here t is that threshold, $\omega_0(t)$ is the number of pixels in which gray value of its image are less than the threshold t. $\omega_1(t)$ is the number of pixels in which the image gray value are greater than the threshold value t. $\mu_0(t)$ is the average gray value of the pixels in which the image gray value are less than the threshold t. And $\mu_1(t)$ is the average gray value of the pixels in which the image gray values are greater than the

threshold T. The t that makes $\sigma(t)^2$ the greatest value is the best partition threshold. Here the best threshold required needs to traverse all the pixel gray values within the scope and calculate the variance and finally arrive at the greatest variance. It is obvious that the calculation cost is huge and the efficiency of calculation is also low; determining the optimal threshold is ultimately an optimization.

3. Edge thinning is performed by computing the gradient magnitude in four directions—horizontal (x1), vertical (x2), 45^0 and 135^0 ([9]-[3]). The four gradients are computed using:

$$I_{x1} = \sqrt{(I(x1, x2) - I(x1, x2-1))^2 + (I(x1, x2) - I(x1, x2+1))^2}$$

$$I_{x2} = \sqrt{(I(x1, x2) - I(x1-1, x2))^2 + (I(x1, x2) - I(x1+1, x2))^2}$$

$$I_{45^0} = \sqrt{(I(x1, x2) - I(x1-1, x2+1))^2 + (I(x1, x2) - I(x1+1, x2-1))^2}$$

$$I_{135^0} = \sqrt{(I(x1, x2) - I(x1-1, x2-1))^2 + (I(x1, x2) - I(x1+1, x2+1))^2}$$

Edges in the edge-image are made one pixel thick by considering a 3×3 neighbouring region around an edge pixel. If any of the following conditions are met, then the pixel is considered to be an edge-pixel, otherwise it is set to zero:

$$I(x1,x2) \geq \max_{(k1,k2)} I(x1+k1,x2+k2) \qquad k_1,k_2 \in \{-1,0,1\}$$

$$I(x1,x2) \geq \max_{(k1,k2)} I(x1+k1,x2+k2) \qquad k_1 = 0, k_2 = \{-1,0,1\} \wedge I_{x1}(x1,x2) \geq I_{x2}(x1,x2)$$

$$I(x1,x2) \geq \max_{(k1,k2)} I(x1+k1,x2+k2) \qquad k_1 \in \{-1,0,1\}, k_2 = 0 \wedge I_{x2}(x1,x2) \geq I_{x2}(x1,x2)$$

$$I(x1,x2) \geq \max_{(k1,k2)} I(x1+k1,x2+k2) \qquad (k_1,k_2) \in \{(-1,-1),(0,0),(+1,+1)\} \wedge I_{45^0}(x1,x2) \wedge I_{135^0}(x1,x2)$$

$$I(x1,x2) \geq \max_{(k1,k2)} I(x1+k1,x2+k2) \qquad (k_1,k_2) \in \{(+1,+1),(0,0),(-1,-1)\} \wedge I_{135^0}(x1,x2) \wedge I_{45^0}(x1,x2)$$

4.2 **Visual Comparison of the Results**

In order to test the effectiveness, we implemented the proposed algorithm in MATLAB 7.0 and the output of this algorithm are compared with the existing algorithms as shown in the following figures.

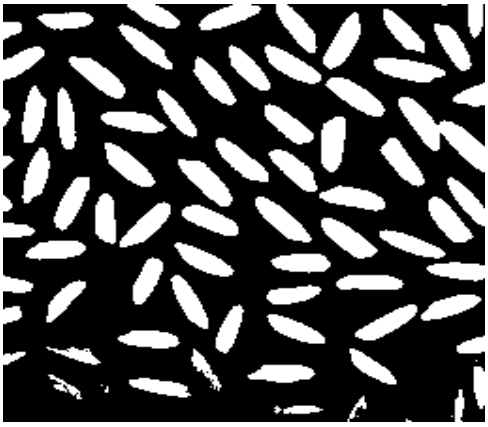


Fig 4.1 Original rice image

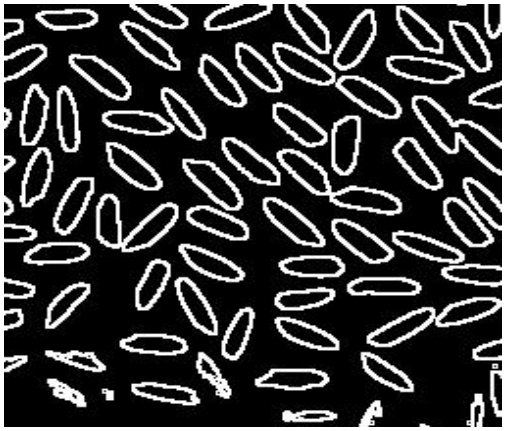


Fig 4.2 Output image Sobel edge detection

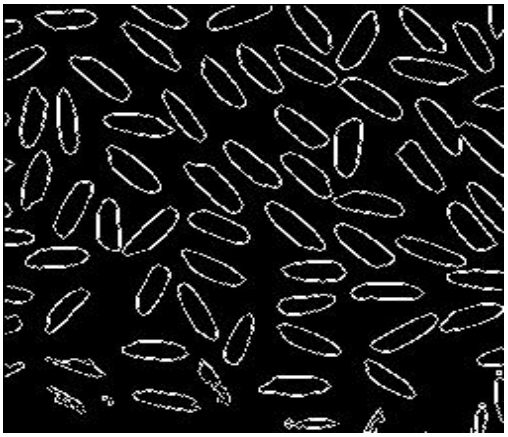


Fig 4.3Output Image of proposed algorithm

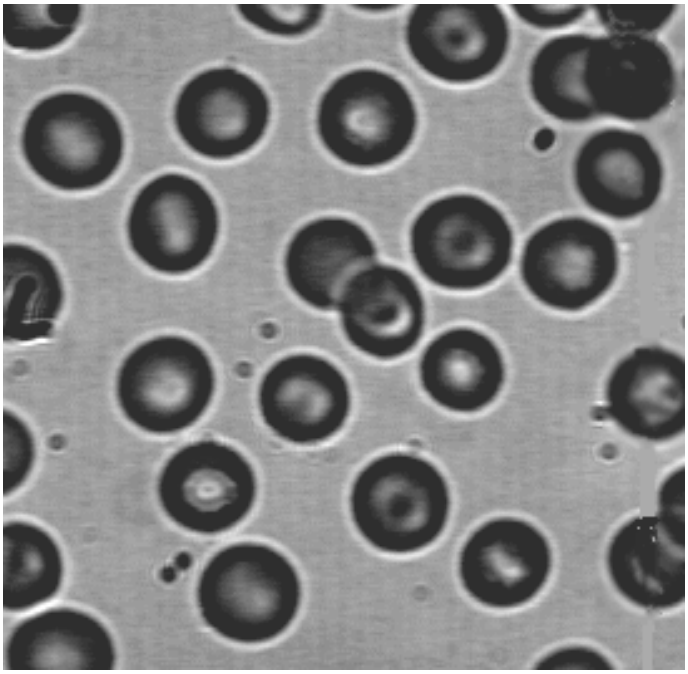


Fig 4.4 Blood original image

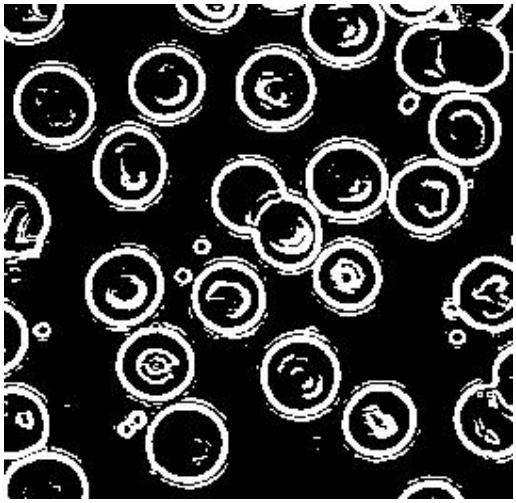


Fig 4.5Output image of Sobel edge detection

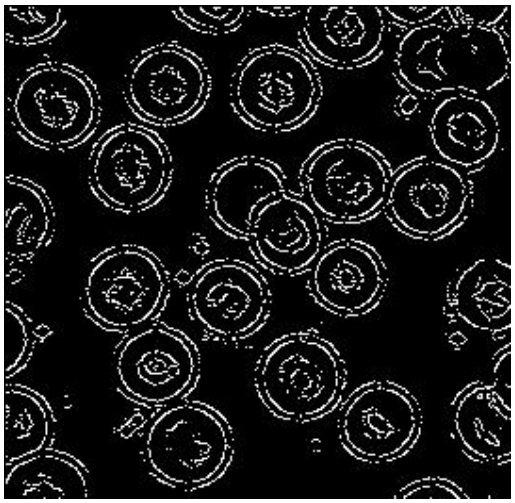


Fig 4.6 Output Image of proposed algorithm

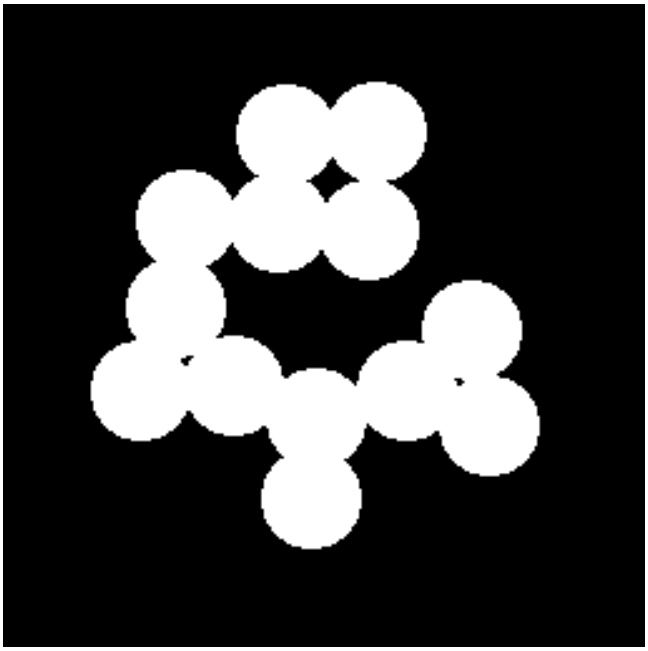


Fig 4.7 Original Circles image

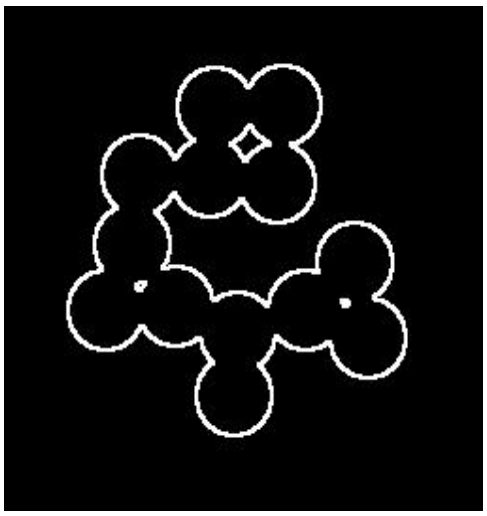


Fig 4.8Output image of Sobel edge detection

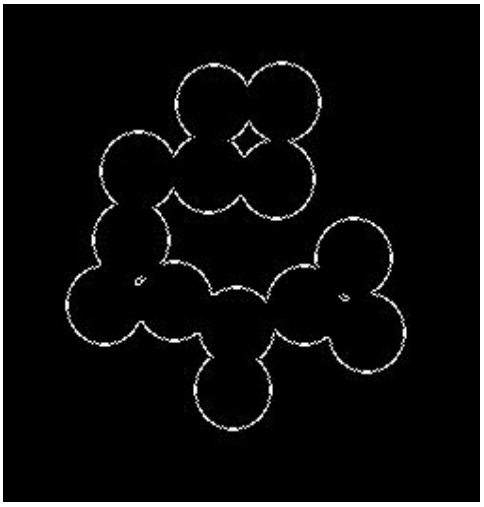


Fig4.9 Output Image of proposed algorithm

Fig 4.10 Original cman image

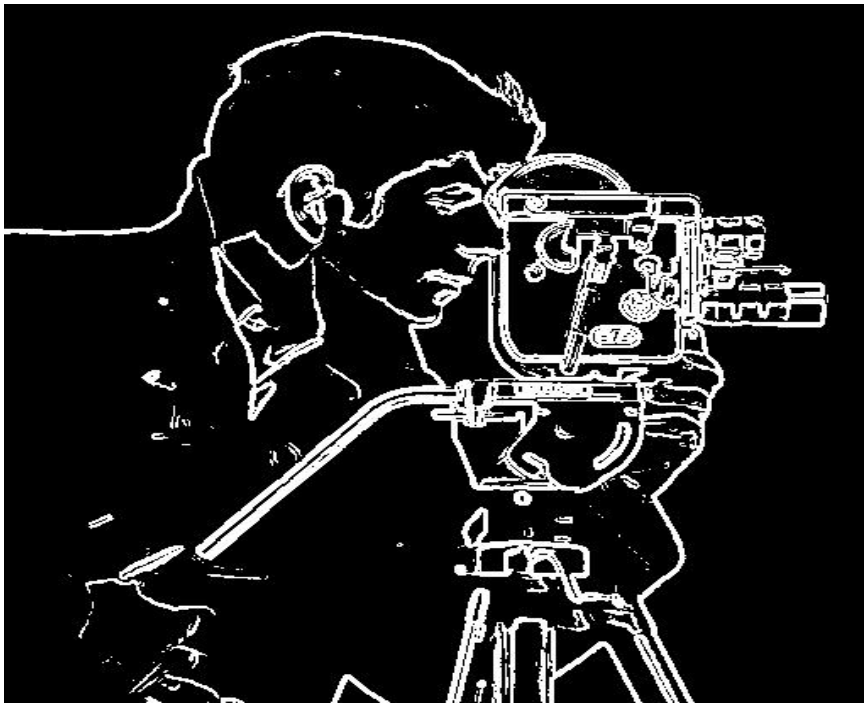


Fig 4.1Output image of Sobel edge detection method

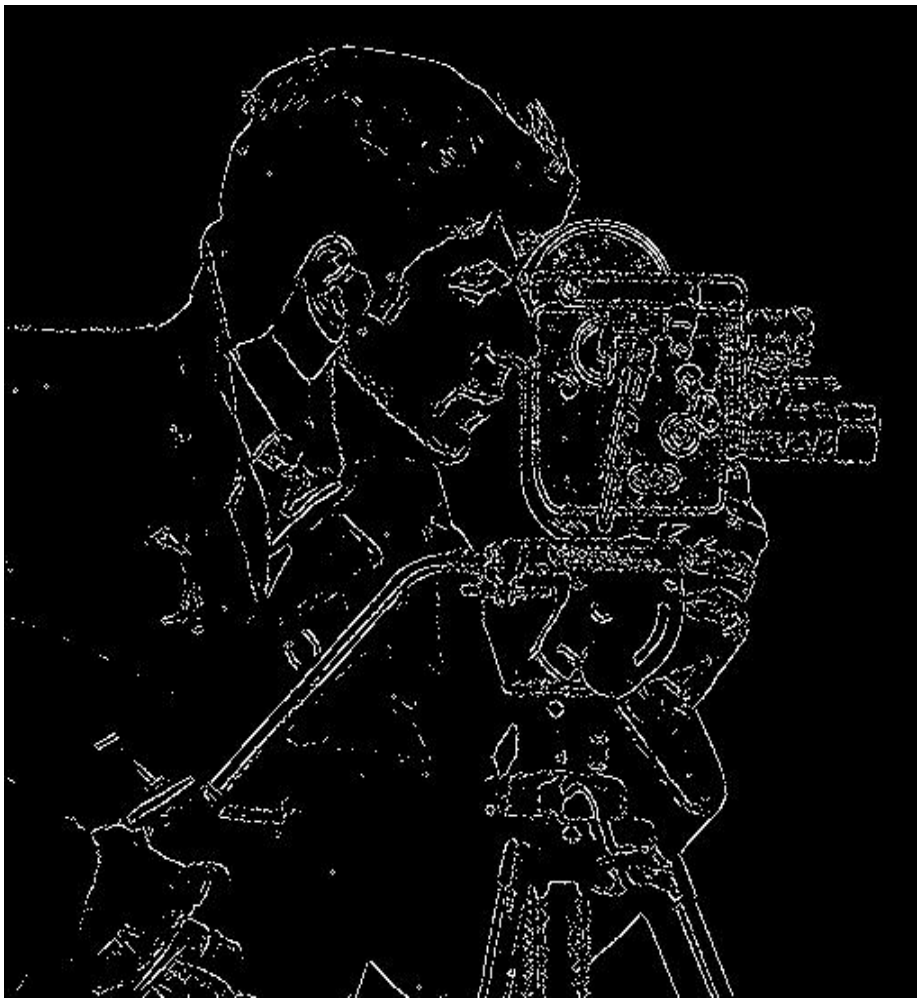


Fig 4.12 Output Image of proposed algorithm

Fig .4.13 Original coins image

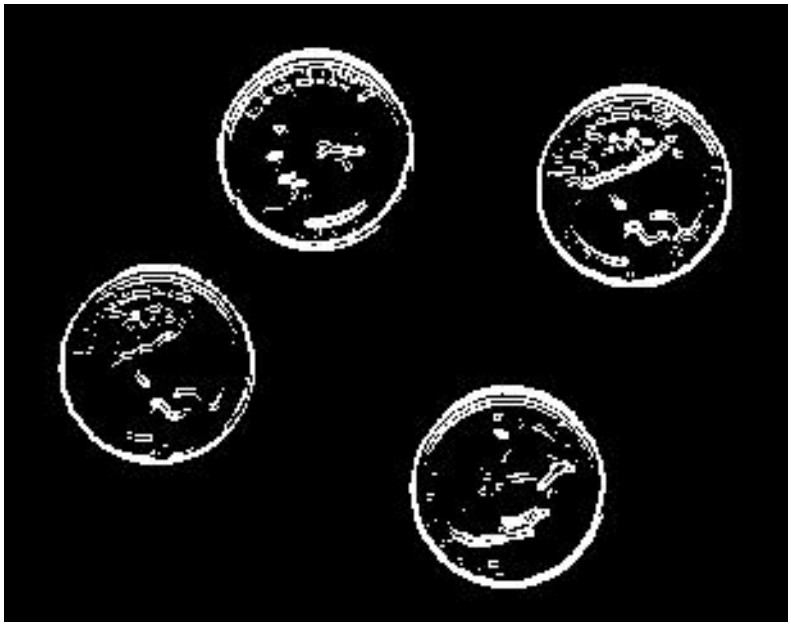


Fig 4.14 Ouput image of Sobel edge detection algorithm

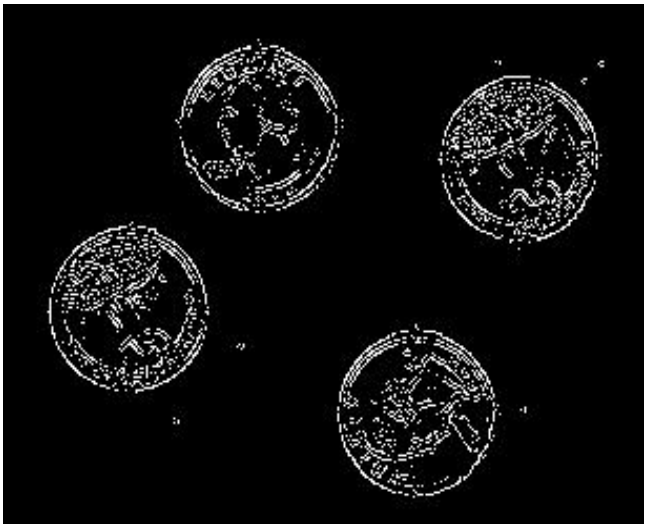


Fig 4.15 Output Image of proposed algorithm

Comparing the results of edge detection of above images in figures (4.3, 4.6, 4.9, 4.12, 4.15) we can easily find that the new algorithm that used improved detection template and algorithm got better result than the traditional Sobel operator. The new algorithm has stronger edge search capability and more complete edge.

Chapter 5

Conclusion and Future work

This thesis presents an improved edge detection method. The output of the proposed method shows that the edges detected are not thick, but they are like the original edges. But there are some also small edges which are not present and some small edges are broken in the output images of this proposed method. The future work will be to study the reasons why these small edges are not present and why they are broken.

Chapter-6

References

[1]	RC Gonzalez and RE Woods, "Digital Image Processing", 3rd Edition, Pearson Education, 2008.
[2]	Zhang Jin-Yu, Chen Yan, and Huang Xian-Xiang, "Edge Detection of Images Based on Improved Sobel Operator and Genetic Algorithms",
[3]	Swathi Tandra, and Ziaur Rahman, "Robust Edge-detection algorithm for runway-edge detection", Old Dominion University, Electrical and Computer Engineering Department Norfolk, Virginia 23529.
[4]	M. Wen and C. Zhong, "Application of Sobel Algorithm in Edge Detection of Images," China High-tech Enterprise, pp.57-62, Jun. 2008.
[5]	X. L. Ci and G. G. Chen, "Analysis and Research of Image Edge Detection Methods," Journal of Infrared, pp. 20-23, July 2008.
[6]	Renyan Zhang, Guoliang Zhao and Li Su, "A New Edge Detection Method in Image Processing", Automation College ,Harbin Engineering University, Harbin, China.
[7]	J. Canny, "A computational approach to edge detection", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 8, pp. 679-698, 1986.
[8]	W. Niblack, "An Introduction to Digital Image Processing". Prentice-Hall International, Denmark, 1986.
[9]	J. Park, H. C. Chen, and S. T. Huang, "A new gray level edge thinning method," in Proceedings of the ISCA 13th International

	Conference on Computer Applications in Industry and Engineering, pp. 114–119,November 2000.
[10]	Prewitt J., “Object Enhancemet And Extraction, Picture Processing and Psychopictorics”, NY, Academic Pres, 1970.
[11]	Ziou D. and Tabbone S., “Edge Detection Techniques – An Overview”, Technical Report, No. 195, Dept. Math & Informatique, Universit de Sherbrooke, 1997.
[12]	R. M. Haralick, “Digital Step Edges from Zero Crossings of Second Directional Derivatives,” IEEE Tran. Pattern Analysis and Machine Intelligence, 6(1), pp. 58—68, January, 1984.
[13]	N. A. Otsu, “A threshold selection method from gray-level histograms,” IEEE Transactions on Systems, Man and Cybernetics, vol.9, no. 1. pp. 62-66, Jan. 1979.
[14]	J. Li, X. K. Tang, and Y. J. Jiang, “Comparing Study of Some Edge Detection Algorithms,” Information Technology, vol.38, no.9, pp. 106-108. Sep. 2007.