CERTIFICATION

This is to certify that this Project Report titled **"IMPLEMENTATION OF CONTRAST ENHANCEMENT OF IMAGE BASE ON HISTOGRAM EQUALIZATION"** was carried out by **BABWALE OPEYEMI TOLULOPE** with the Matriculation Number **CSC/11/O273** in partial fulfilment for the award of Bachelor of Science Degree in Computer Science, Federal University Oye Ekiti, Ekiti State, Nigeria.

---------- 22-10-15

Mr Fagbuagun O. A **DATE**

Supervisor

---------- 26/10/2015

Dr Oluwadare. S.A **DATE**

Ag. Head of Department

DEDICATION

I dedicate this project first and foremost to Almighty God who has been there right from the beginning to this very point. Special dedication also to my ever supportive folks, for their relentless support and compassion towards me during the course of my four years in the university. A big thanks to all the staff in the Department of computer science (FUOYE) and my best friend.

To God is the glory.

ACKNOWLEDGEMENT

I hereby express my profound gratitude to God Almighty who created me and to my wonderful mother Mrs Idowu Funmilayo Babawale for her love, care, encouragement, moral and financial support throughout my stay in the university.

Special thanks to my Supervisor Mr. Fagbuagun O. A for his thorough supervision throughout this project work and also to Primate Dr. S.O Ojuolape for his financial and spiritual support throughout my stay in the university, indeed he is a father.

My special thanks also goes to Dr. Adebayo Adetunmbi (Head of Department) and the entire staff of the department of computer science Federal University Oye Ekiti, Ekiti State for their direction and to my Siblings Tola, Iyiola, Dipo, and Precious for their financial and moral support love you guys. I also say big thanks to my uncle Obamina Johnson.

Finally, I thank each and everyone that helped to complete this project work with their cordial support most especially my best friend and all the CSC FUOYE pioneers.

TABLE OF CONTENTS

CHAPTER FIVE – SUMMARY, CONCLUSION AND RECOMMENDATION

LIST OF FIGURES

ABSTRACT

Contrast enhancement is a critical step in Image processing and segmentation for computer vision. It can be done using several methodologies among which are histogram equalization and edge enhancement. This project work is based on contrast enhancement by the method of histogram equalization in which the knowledge of the histogram is used to improve the resolution of the image. The input to the system are images taking with digital camera of known resolution. The images are processed using MATLAB and the result are shown as well. It was discovered that histogram equalization makes image segmentation an easy task in computer vision.

CHAPTER ONE

1.1 Background of the study

Image segmentation, a process of pixel classification, aims to extract or segment objects, or regions from the background. It is a critical processing step to the success of image recognition. There is no single standard approach to segmentation because many different types of scene parts can serve as the segments on which descriptions are based, and there are many different ways in which one can attempt to extract these parts from the image. Selection of an appropriate segmentation technique depends on the type of images and applications. Image segmentation algorithms are based on two basic properties of the pixel intensities in relation to their local neighborhood: discontinuity and similarity. Segmentation methods based on pixel discontinuity are called boundary-based or edge extraction methods, whereas methods based on pixel similarity are called region-based methods. Boundary-based methods assume that the properties, such as intensity, color, and texture (Khotanzad & Chen, 1989; Panjwani & Healey, 1995), should change abruptly between different regions. Region-based methods assume that neighboring pixels within the same region should have similar values (e.g., intensity, color, and texture). The evaluation of segmentation algorithms has been mostly subjective; therefore, people usually judge an algorithm's effectiveness based on intuition and the results from several segmented images (Frank, 2010). This project work is intended to implement a chosen segmentation algorithm and present the result on different types of image scenes.

1.2 Statement of problem

Image segmentation is an important step in making a computer to recognize images called image recognition. The main goal is to divide an image into parts that have a strong correlation with objects or areas of the real world contained in the image. However, there are challenges to image

segmentation such as poor contrast, presence of shadows which can affect clustering of the data, and noise, which is an unwanted part of the image and must be removed. This project work deals with how to overcome these challenges and make segmentation of an image an efficient task.

1.3 Purpose of Study

In this project, we investigate uses of histograms in greyscale image processing. The codes, as well as images, and figures created from this project were accomplished using Matrix Laboratory (Matlab). An image histogram shows the frequency of occurrence each allowable pixel value has in the image. These frequencies will be manipulated in segmentation process. For the greyscale images used in this project, these were integer values on the range [0,255], where 0 represents pure black and 255 represents pure white. Values in between are varying shades of grey.

1.4 Objectives of Study

The objectives of the study are:

(i) to construct a histogram from a grayscale image

(ii) to perform histogram equalization on the image

1.5 Scope of Study

There are several image segmentation methods available. One of the simplest approaches to segment an image is based on the intensity levels and is called as threshold based approach. Another is edge based segmentation. We also have region based segmentation and clustering based segmentation methods. This project work only covers the area of histogram equalization and does not seek to implement all the methods identified above

1.6 Significance of Study

Choosing a threshold value for image segmentation is not an easy task because different images have different contrasts and different amount of noise and this limit the generality of the

threshold value. The major problem with thresholding is that only the intensity is considered, not any relationships between the pixels. There is no guarantee that the pixels identified by the thresholding process are contiguous. We can easily include extraneous pixels that aren't part of the desired region, and we can just as easily miss isolated pixels within the region (especially near the boundaries of the region). These effects get worse as the noise gets worse, simply because it's more likely that a pixel's intensity doesn't represent the normal intensity in the region. Therefore, it is necessary to develop a means of choosing a threshold value that properly will segment an image into background and foreground. This is the main point of this project work.

1.7 Limitation of Study

The images used in this project work are taken with digital camera of a known resolution. They are not taken in motion and therefore motion pictures are not involved. Images from video are not also included in the project work as they require sophisticated image processing techniques to segment them due the element of motion involved.

1.8 Research Methodology

In this project, the use of histograms in greyscale image processing is investigated. All code, images, and figures created from this project were accomplished using Matlab. The first goal was to be able to construct a histogram from a greyscale image. An image histogram shows the frequency of occurrence each allowable pixel value has in the image. For the greyscale images used in this project, these were integer values on the range [0,255], where 0 represents pure black and 255 represents pure white. Values in between are varying shades of grey. The second part of the project was to perform a process known as histogram equalization which increases contrast of an image by effectively stretching the histogram, or "equalizing" the frequencies. This works well with images that have a low range of greyscale values as they can be stretched out further.

The final part of this project deals with image segmentation. The work is organized in chapters. Chapter one deals with introduction. Chapter two is the review of literature work while chapter three is the research methodology. Chapter four deal with report while chapter five is the summary, conclusion and recommendation.

1.9 Definition of terms

Image: An image is a rectangular grid of pixels. It has a definite height and a definite width counted in pixels.

Threshholding: Thresholding creates binary images from grey-level ones by turning all pixels below some threshold to zero (0) and all pixels about that threshold to one (1).

Binary image: A binary image is a digital image that has only two possible values for each pixel. Typically the colour used for the objects in an image is the foreground colour white while the rest of the image is the background colour.

Grayscale image: Grayscale image is an image that has a range of shades of gray without apparent color. The darkest possible shade is black, which is the total absence of transmitted or reflected light. The lightest possible shade is white, the total transmission or reflection of light at all visible wavelength. Black and white image has only two colours, black and white.

Cluster: A computer cluster consists of a set of loosely or tightly connected computers that work together so that, in many respects, they can be viewed as a single system. Clustering in data analysis is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups. It is used in image analysis.

Segmentation: Segmentation involves separating an image into regions (or their contours) corresponding to objects.

Pixel: in digital imaging, a pixel (also known as picture element) is a physical point in a raster image, or the smallest addressable display device.

Resolution: Resolution is the quality of an image. It is the detail an image holds. It is determined by the size of the units of information representing an image. This unit of information is the pixel.

Edge: An edge in an image is a boundary or contour at which a significant change occurs in some physical aspect of an image, such as illumination or surface reflectance.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

Segmentation refers to the process of partitioning a digital image into the multiple segments (set of pixels as known as super pixels). The goal of segmentation is to simplify and or change the representation of an image into something that is more meaningful and easier to analyze (Raju and Neelima, 2012). The goal of image segmentation is to cluster pixels into important image regions corresponding to individual surfaces, objects, or natural parts of objects. Regions are usually segmented by identifying common properties of an image. Also, contours can be identified by finding the difference between regions of the image. The pixels in a region in an image share a common property known as pixels intensity. Therefore, a natural way to segment such regions is through image thresholding, the separation of light and dark regions. Thresholding creates binary images from grey-level ones by turning all pixels below some threshold to zero and all pixels about that threshold to one. In this research work, I will look into the properties of an image, the definition of image and how it can be represented before manipulation.

2.2 Digital Image definition

According to Young, Gerbrands and Vliet (2007), a digital image $a[m,n]$ described in a 2D discrete space is derived from an analog image $a(x,y)$ in a 2D continuous space through a *sampling* process that is frequently referred to as digitization. An image is nothing more than a two dimensional signal. It is defined by the mathematical function a(x,y) where x and y are the two co-ordinates horizontally and vertically. The value of a(x,y) at any point is gives the pixel value at that point of an image.

Going by this definition, image can be in two forms: Continuous and discontinuous forms. The continuous form of representing image is the analog form of an image while the discontinuous form is the digital form. Because analog signal requires a lot of memory to store and manipulate it, images are usually processed in digital form. In this case, the values of the image had been discretized. Mathematically, the analog image is designated as a(x,y) while the digital form is designated as a[x,y].

An image is shown in figure 2.1 below shows a digital image.

Figure 2.1: A digital image

The above image is nothing but a two dimensional array of numbers ranging between 0 and 255 as shown below in equation 2.1.

$$f[x,y] = \begin{bmatrix} 128 & 30 & 123 \\ 232 & 123 & 123 \\ 123 & 77 & 89 \\ 80 & 255 & 255 \end{bmatrix} \quad \text{.....(1)}$$

Each number represents the value of the function f(x,y) at any point. In this case the value 128, 230, and 123 each represents an individual pixel value. The dimensions of the picture is actually the dimensions of this two dimensional array.

2.3 Analog and digital image processing

Analog image processing is done on analog signals. It includes processing on two dimensional analog signals. In this type of processing, the images are manipulated by electrical means by varying the electrical signal. The common example include is the television image. Digital image processing has dominated over analog image processing with the passage of time due its wider range of applications. It deals with the development of digital system that performs operation on a digital image.

2.4 Types of digital images

Binary Image: In a binary image, each pixel is just black or white because there are only two possible values for each pixel. Binary images are very efficient in terms of storage and they are suitable for text, fingerprints or architectural plans (McAndrew, 2004).

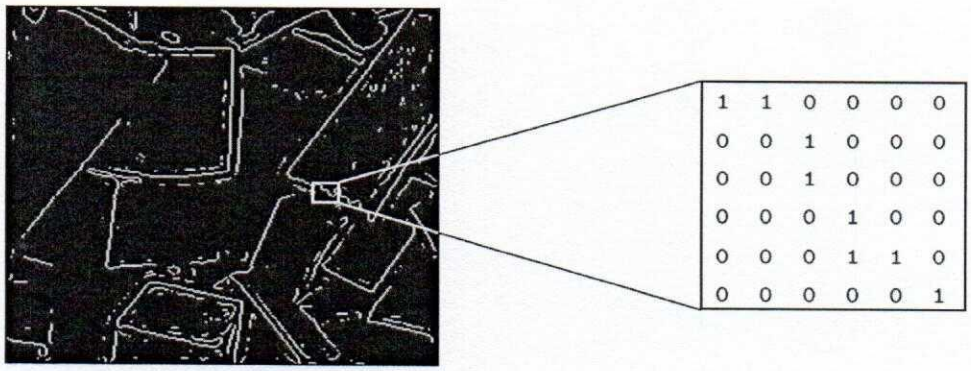


Figure 2.2: A binary image. Source: Gonzales and Woods, 2002

Grayscale image:

Each pixel is a shade of grey, normally from 0 (black) to 255 (white). This range means that each pixel can be represented by eight bits, or exactly one byte. This is a very natural range for image file handling. Other greyscale ranges are used, but generally they are a power of 2. Such images arise in medicine (X-rays), images of printed works, and indeed 256 different grey levels is

sufficient for the recognition of most natural objects (McAndrew, 2004). Figure 2.3 below shows an example of a grayscale image.

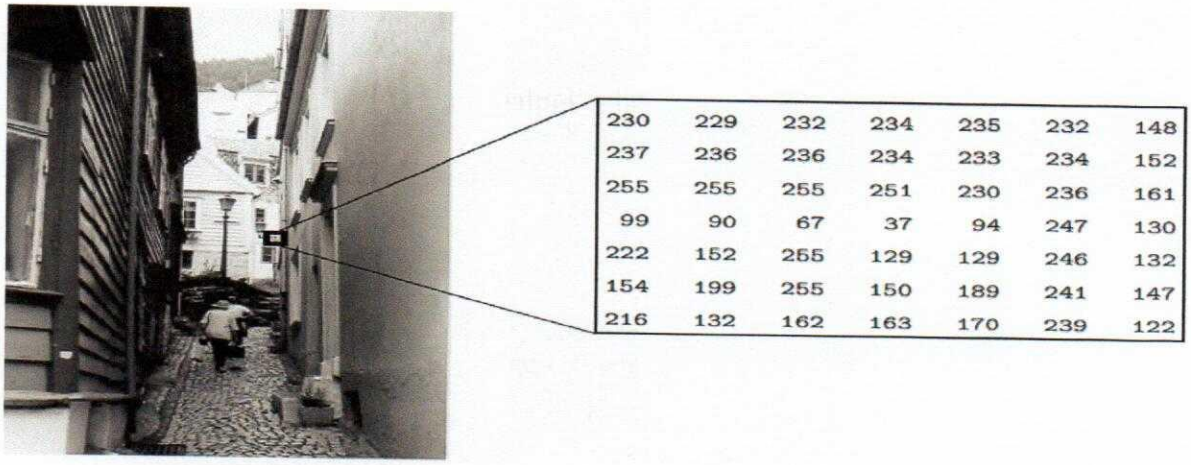


Figure 2.3: A grayscale image of a street. Source: Gonzales and Woods, 2002

RGB Image (True colour): In (McAndrew, 2004), RGB (Red, Green, Blue) image is an image in which each pixel has a particular colour, which I being described by the amount of red, green and blue in it. Each of these components has a range of 0-255 intensity values in it which gives a total of 255^3 or 16,777,216 different possible colours in the image. This type of image is considered as consisting of a stack of three matrices, representing the red, green and blue values for each pixel. An example is shown in figure 2.4 below.

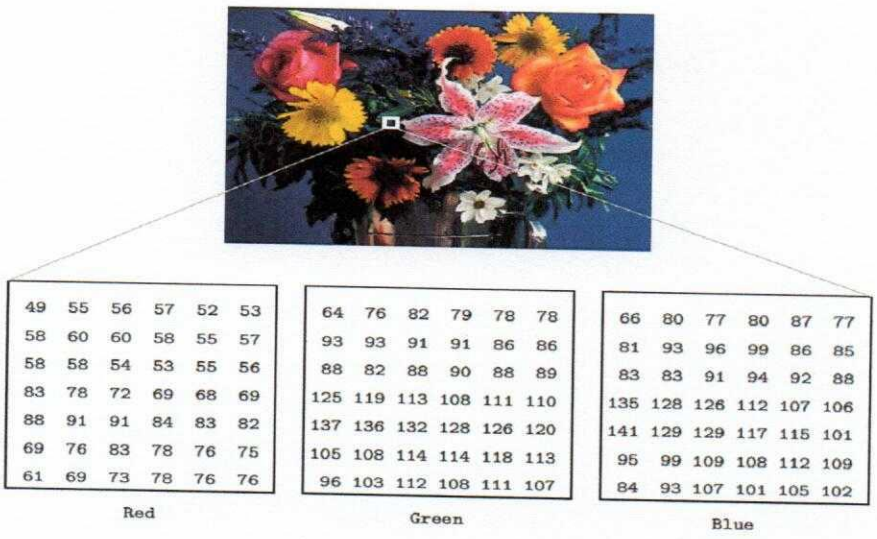


Figure 2.4: A true colour image. Source: Gonzales and Woods, 2002

In this project work, images used may be transformed from one type to another to suit the image processing task performed at each stage of the implementation of the project. However, colour images are not used throughout the project work. They only exist as the source image and once they are transformed to grayscale, they are not reversed back to true colour again. In fact, binary and grayscale images are mostly used in the project work.

2.5 Image intensity

Image intensity is described in terms of the amount of light falling on a surface or the reflected light. Traditionally, image intensities have been processed to segment an image into regions or to find edge fragments. Image intensities carry a great deal more information about three-dimensional shapes and can be used to extract information from the image.

2.6 Image processing techniques

Digital image processing focuses on two major tasks: One, the improvement of pictorial information for human interpretation; and two, processing of image data for storage, transmission and representation for autonomous machine perception. In image processing operation are performed on images that transform it from one representation to the other (Young and Gerbrands, 2007). He stated further that the types of operations that can be applied to digital images to transform an input image $a[m,n]$ into an output image $b[m,n]$ (or another representation) can be classified into three categories:

(i) Point Operations

(ii) Local operations and

(iii) Global operations

Point operations:

Before point operations are applied to images, the image has to be sampled, quantized and stored in digital form. Some image processing operations such as contrast enhancement, extraction of

one or more information from the data can be carried out. A **point operation** on an image is an algorithm that changes each pixel value according to some function:

$$I(u,v) \mapsto f(I(u,v)) \quad \text{.....(2)}$$

The function f depends only on the pixel value and it is independent of the spatial location (u; v). The domain of f must match the range of I, and the range of f determines the output image type. Examples of point operations are:

(i) Addition: this involves changes in brightness of an image. A constant value is added to each and every pixel on the input image to increase the contrast. This is done according to the equation (3) below

$$f(p) = p + k \quad \text{.....(3)}$$

where k is the constant value added.

(ii) Multiplication: this point operation stretches or shrinks the image range and can be represented in equation (4) below $f(p) = p * k$ .....(4)

The various operators O can be grouped based on the number and location of pixels of the input image f that affect the computation of a particular output pixel g [x, y].

Point Operations on single images: The gray value of the output image g at a particular pixel [x, y] depends ONLY on the gray value of the same pixel in f ; examples of these operations include contrast stretching, segmentation based on gray value, and histogram equalization;

Point Operations on multiple images: The gray value of the output pixel g [x, y] depends on the gray values of the same pixel in a set of input images f [x, y, tn] or f [x, y, λn]; examples are

segmentation based on variations in time or color; multiple-frame averaging for noise smoothing, change detection, and spatial detector normalization;

Neighborhood Operations on one image: The gray value of g at a particular pixel [x, y] depends on the gray values of pixels in the neighborhood of the same pixel in f [x, y]; examples include convolution (as for image smoothing or sharpening), and spatial feature detection (e.g., line, edge, and corner detection).

Neighborhood Operations on multiple images: This is just a generalization of neighborhood operation on multiple images; the pixel g [x, y] depends on pixels in the spatial and temporal (or spectral) neighborhood of [x, y, tn or λn]. Spatial / temporal convolution or spatial / spectral convolution.

Operations based on Object "Shape" (e.g., "structural" or "morphological") operations: The gray level of the output pixel is determined by the object class to which a pixel belongs; examples include classification, segmentation, data compression, character recognition;

Geometrical Operations: The pixels f [x, y] are remapped to a new coordinate system to obtain g [x, y]: image warping, cartography;

"Global" Operations: The gray value of the output image at a pixel depends on the gray values of all of the pixels of f [x, y]; these include image transformations, e.g., Fourier transform, Hartley transform, Hough transform, Haar transform and Radon transform.

2.6 Image noise

Noise is an unwanted part of a signal and may be present in an image. Seema and Meenakshi (2013) stated that noisy image consist of unwanted data which may reduce the shape and size of objects in the image and blurring of edges or dilution of fine details in the image so eliminating such noise is an important pre-processing task. Therefore it is desirable to remove noise since they corrupt image and may lead to false signal being detected in images. The process of removing noise is called de-noising. Denoising is one of the important pre-processing tasks for

various image processing. Image noise is the random variation of brightness and color information in images produced by the scanner and digital camera.

Images can be corrupted during data transmission due to electromagnetic interference in the transmission channel. When image transmission is done through a wireless network, lightening and some atmospheric conditions such as temperature might affect the image transmitted (Srinivasan & Ebenezer, 2007). According to Singh and Prakash (2012), an image degraded by noise can be modeled as shown in equation 5 below.

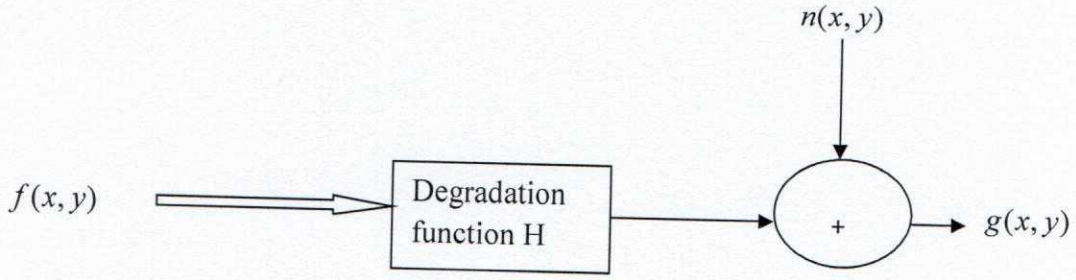


Figure 2.7: Model of image degradation

$$g(x,y) = h(x,y) \otimes f(x,y) + n(x,y) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

Where $g(x,y)$ is the degraded image; $h(x,y)$ is the spatial representation of the degradation function; $f(x,y)$ is an original image and $n(x,y)$ is noise. The symbol "$\otimes$" indicate the convolution which means a function is being slided over another function.

2.7 Types of image noise

Image noise can be divided to two types: external and internal noise. External noise, means that the system or external electromagnetic interference to the string into the power caused by noise within the system. External noise include: electrical equipment, celestial phenomena caused by the discharge noise (Image Noise of classifications in Remote Sensing, 2011).

Internal noise in general can be divided into the following four types:

(i) The basic properties of light caused by electrical noise. Like the current generation is a collection of particles electrons or holes. Because these particles in the random formation of shot noise; conductor free electrons irregular thermal motion of the formation of thermal noise.

(ii) Electrical noise generated by the mechanical movement. Jitter caused by a variety of joints such as the current change due to the noise generated; heads, tape, etc.

(iii) Equipment, material itself due to noise. Such as positive and negative particles and the surface of the disk surface defects produced by tape noise. With the development of materials science, it expected to continue to reduce the noise.

(4) Equipment within the system caused by circuit noise. It has the introduction of the AC power supply noise; deflection system and clamp circuit caused by the noise (Image Noise of classifications in Remote Sensing, 2011).

In this project work, the following types of image noise will be reviewed because they are relevant to the study at hand.

2.8 Noise Modeling

Noise may be modeled either by a histogram or by a probability density function (p.d.f.) which is superimposed on the probability density function of the original image. In the models below, the most common types of noise will be presented as they are applicable to this project work.

2.8.1 Gaussian Noise

This type of noise is also called the Gaussian noise or normal noise is randomly occurs as white intensity values (Salem, Kalyankar and Khamitkar, 2010). The Gaussian noise has a normal probability density function. It is a statistical noise having a probability density function equal to that of the normal distribution, which is also known as the Gaussian distribution. This also means that the values that the noise can take on are Gaussian-distributed. The probability density function p of a Gaussian random variable z is given by the equation 6 below.

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \quad \text{.....(6).}$$

Where z represent the grey level, μ represent the mean value and σ represent the standard deviation of the image.

The principal source of Gaussian noise in digital images occurs during image acquisition such as sensor noise that is caused by poor illumination or high temperature or the transmission medium.

2.8.2 Salt and pepper noise

This type contains random occurrences of both black and white intensity values, and often caused by threshold of noise image (Salem, Kalyankar and Khamitkar, 2010). Therefore, the presence of single dark pixels in bright regions, or single bright pixels in dark regions, is called salt and pepper noise. Salt and pepper noise is the natural result of creating a binary image via thresholding. Salt corresponds to pixels in a dark region that somehow passed the threshold for bright, and pepper corresponds to pixels in a bright region that were below threshold. Salt and pepper might be classification errors resulting from variation in the surface material or illumination, or perhaps noise in the analog/digital conversion process in the frame grabber.

The probability density function for salt and pepper noise is as shown in figures 2.5.

$$p(z) = \begin{cases} P_a & for\ z = a \\ P_b & for\ z = b \\ 0 & otherwise \end{cases}$$

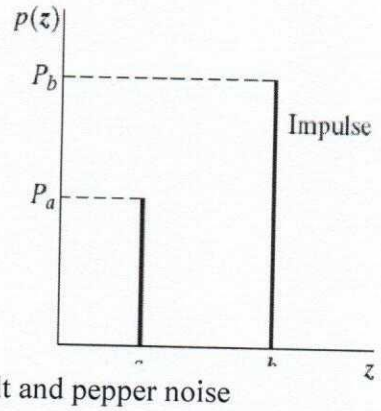


Figure 2.5: Probability density function of salt and pepper noise

2.8.3 Uniform Noise

The uniform noise cause by quantizing the pixels of image to a number of distinct levels is also known as quantization noise. It has approximately uniform distribution. In the uniform noise the level of the gray values of the noise are uniformly distributed across a specified range. the probability density function of uniform noise is given by

$$p(z)=\begin{cases}\frac{1}{(b-a)} & \text{if } a\le z\le b\\ 0 & \text{otherwise}\end{cases}$$

$$\mu=(a+b)/2; \qquad \sigma^2=(b-a)^2/12 \qquad \text{.....(7)}$$

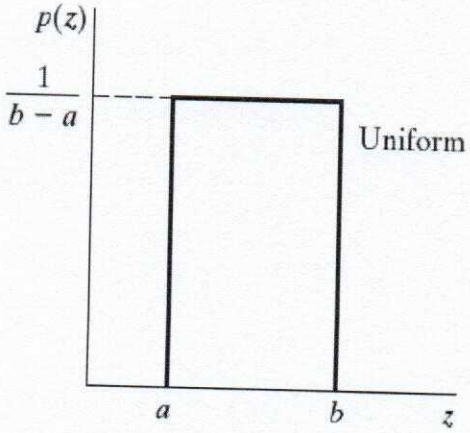


Figure 2.6: Probability density function, mean and variance of uniform noise

2.8.4 Speckle Noise

Gaussian noise can be modeled by random values added to image but speckle noise can be modeled by random values multiplied by pixel values, it is also called multiplicative noise Gonzales and Woods (2002). Speckle noise is a major problem in some radar applications. Although Gaussian noise and speckle noise appear superficially similar, they are formed by two

totally different methods and thus require the different approaches for their removal. Figure 2.7 shows the effect of speckle noise on an image.

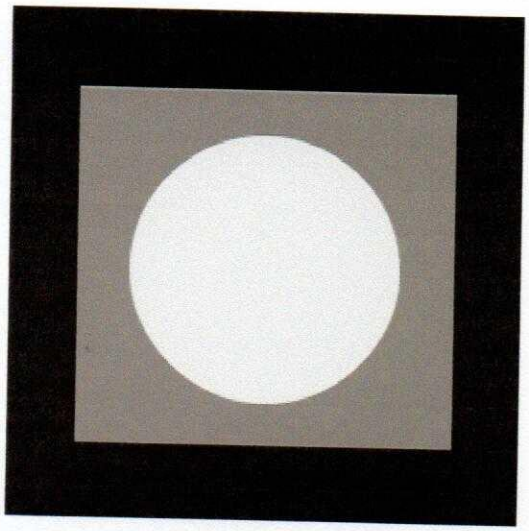


Figure 2.7 a: Original image

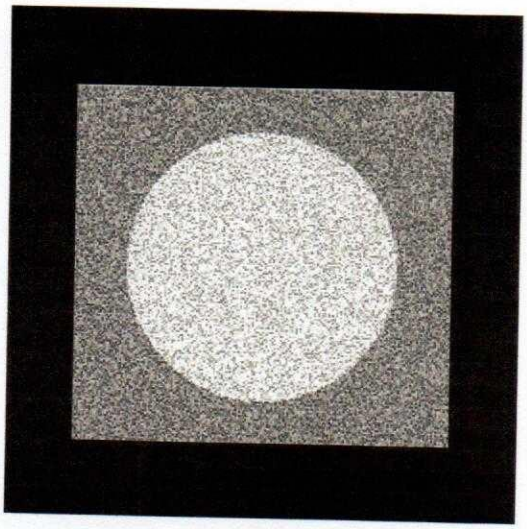


Figure 2.7 b: Image with speckle noise

2.9 Image de-noising techniques

Image denoising, otherwise called image smoothing or image filtering, is the process of removing noise from an image. Noise is an unwanted signal in images and therefore need to be removed. Various techniques are available for removing various models of noise. Dots on image can be modeled as impulses (Salt and pepper noise or speckle) or continuously varying as in Gaussian noise. Speckles in image can be removed by taking the mean or median values of neighbouring pixels (e.g. 3x3 window). This is equivalent to low-pass filtering. The problem with low-pass filtering is that edges may be blurred. More advanced techniques may be used such as adaptive filtering and edge preserving filtering.

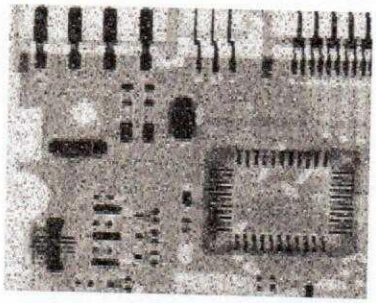


Figure 2.8a: Image with salt and pepper

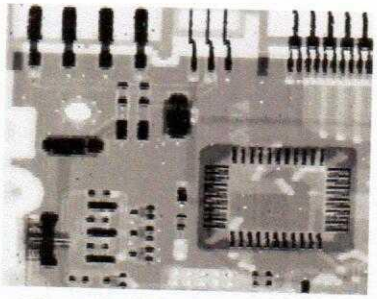


Figure 2.8b: Noise reduction with 3x3 filter.

Mean filters: The mean filter is a non-linear filter, which can be used to remove noise from an image. A mean filter find the average of the current pixel and its neighbours and use the average to replace the current pixel in the output image. It reduces the amount of intensity differences between one pixel and the next. The effect of this is that it removes pixel values that do not represent their surroundings. The problem with averaging filter is that the edges and details of an image are blurred and it is not effective for removing impulse noise (salt and pepper noise).

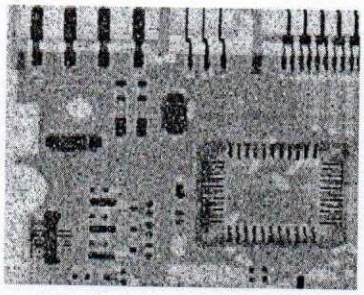


Figure 2.9a: Image with salt and pepper

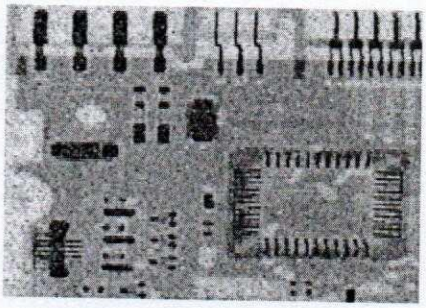


Figure 2.9b: noise reduction with 3x3 mean filter

Median filters: The problem with averaging filters is that they blur edges and they are not effective for impulse noise (salt and pepper). The median filter takes the median value instead of the average or weighted average of pixels in the chosen window. The median filter uses the ordered statistic by selecting the middle value and replacing the output pixel with it. This is done by sorting the value from low to high, and taking the middle value. In a situation whereby we have two values as the middle values, the average of the two values is taken as the median. This type of filter can be used to remove salt and pepper noise from an image. It gives a better result to remove salt and pepper noise in that it completely removes the noise. With an average filter, the colour value of the noise particles are still used in the calculation of the average, therefore when taking the median, all that is done is keeping the colour value of one of two pixels without noise.

Linear filters: These are filters applied by convolution, which is a linear operation, with a low pass filter convolution kernel. According to Kim and Kasper (2013), convolution can be intuitively described as a function that is the integral or summation of two component functions, and that measures the amount of overlap as one function is shifted over the other. An easy way to think of convolution with respect to one variable is to picture a square pulse sliding across the x-axis towards a second square pulse. The convolution at a point is the product of the two functions that occurs when the leading edge of the moving pulse is at that point. Convolution can be used to remove Gaussian noise.

Weighted Averaging filters

Instead of averaging all the pixel values in the chosen window, we can give the closer-by pixels higher weighting, and far-away pixels lower weighting.

$$g(m,n) = \sum_{l=-L}^{L} \sum_{k=-L}^{L} h(k,l)s(m-k,n-l) \quad \text{.....(8)}$$

This type of operation for arbitrary weighting matrices is generally called "2-D convolution or filtering". When all the weights are positive, it corresponds to weighted average. Weighted average filter retains low frequency and suppresses high frequency. This is low-pass filter.

2.10 Edge detection

Omwoyo, Magana, Mazana, Maguya and Márquez (2007) defined edge detection as the process of finding sharp contrasts in the intensities of an image. Edge detection reduces the amount of data in an image while preserving important structural features of that image. Edges are places with strong intensity contrast and they represent object boundaries and therefore can be used in image segmentation to subdivide an image into its constituent regions or objects.

Edge detection refers to the process of identifying and locating sharp discontinuities in an image. The discontinuities are abrupt changes in pixel intensity which characterize boundaries of objects in a scene. The discontinuities can be detected and quantified by finding the local maxima of the first order derivative of the image intensity, that is, the gradient. It can also be detected by finding the zero crossings of the second order derivative of the image intensity, that is, the Laplacian (Rangarajan, 2010). These techniques will be discussed into details under the techniques of edge detection below.

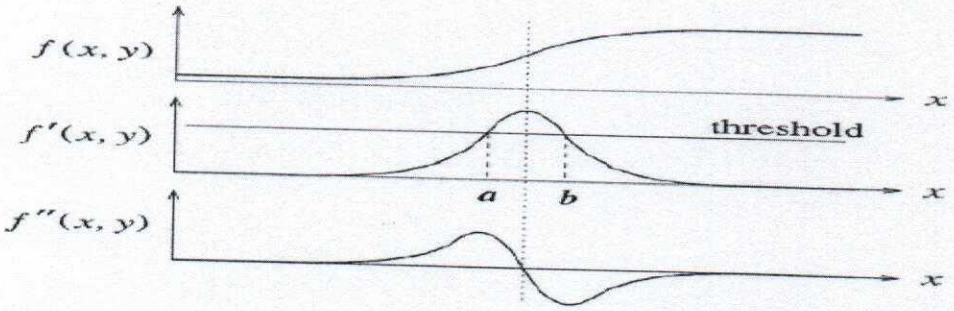


Figure 2.5: First (gradient) and second (Laplacian) derivative edge detection methods

2.11 Edge detection Techniques

Classical methods of edge detection involve convolving the image with an operator (a 2-D filter), which is constructed to be sensitive to large gradients in the image while returning values of zero in uniform regions. There is an extremely large number of edge detection operators available, each designed to be sensitive to certain types of edges. Variables involved in the selection of an edge detection operator include:

(i) **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.

(ii) **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 (Rangarajan, 2010).

(iii) **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 example, edges associated with hair from edges associated with a face.

According to (Rangarajan, 2010), there are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories:

(i) **Gradient method:** The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image.

(ii) **Laplacian method:** The Laplacian method searches for zero crossings in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location.

For example, given the signal in figure 2.10,

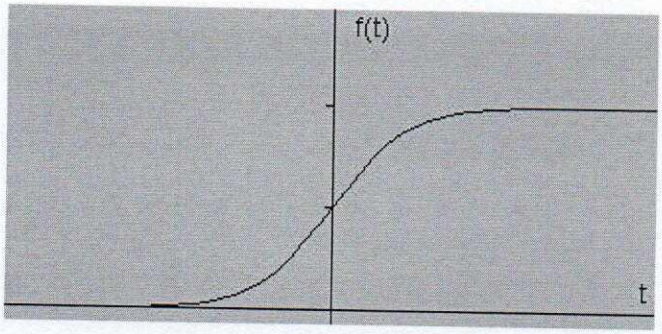


Figure 2.10: An image signal in 2-Dimension t and f(t).

The gradient of the function f(t) is the derivative with respect to t, which is depicted in figure 2.11.

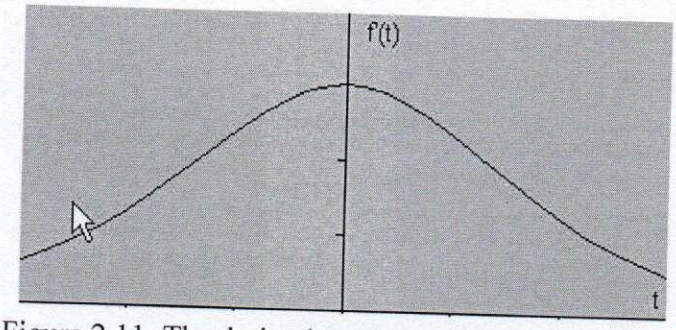


Figure 2.11: The derivative of signal f(t) with respect to

From the derivative, the maximum is located at the center of the edge in the original signal. This local maximum denotes the presence of an edge. This method of locating edges is common to gradient filter family of edge detection filters. This includes Sobel edge detection method.

A pixel location is declared an edge location if the value of the gradient exceeds some threshold. Edges will have higher pixel intensity values than those surrounding it. So once a threshold is set, we can compare the gradient value to the threshold value and detect an edge whenever the threshold is exceeded. Furthermore, when the first derivative is at a maximum, the second derivative is zero. As a result, another alternative to finding the location of an edge is to locate the zeros in the second derivative. This method is known as the Laplacian. The second derivative of the signal is shown in figure 2.12.

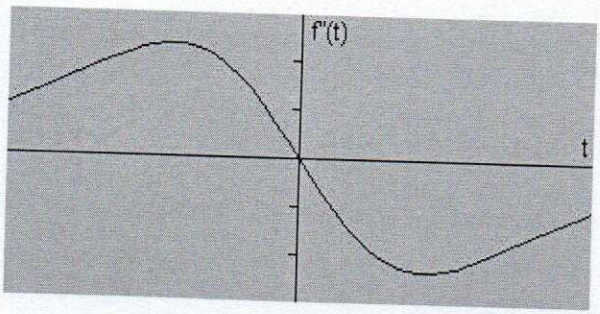


Figure 2.12: The Laplacian of f(t)

2.12 Canny Edge detector

The purpose of edge detection in general is to significantly reduce the amount of data in an image, while preserving the structural properties to be used for further image processing. The Canny Edge Detector is one of the most commonly used image processing tools, detecting edges in a very robust manner.

Canny Edge Detection Algorithm

The Canny edge detection algorithm is known to many as the optimal edge detector. Canny's intentions were to enhance the many edge detectors already out at the time he started his work. He was very successful in achieving his goal and his ideas and methods can be found in his paper, "*A Computational Approach to Edge Detection*". In his paper, he followed a list of criteria to improve current methods of edge detection. Canny (1986) operator is based on three criteria.

(i) The first and most obvious is low error rate. It is important that edges occurring in images should not be missed and that there be NO responses to non-edges.

(ii) The second criterion is that the edge points be well localized. In other words, the distance between the edge pixels as found by the detector and the actual edge is to be at a minimum.

(iii) A third criterion is to have only one response to a single edge. This was implemented because the first 2 were not substantial enough to completely eliminate the possibility of multiple responses to an edge.

Smoothing (Noise removal): Based on these criteria, the canny edge detector first smoothes the image to eliminate noise. It then finds the image gradient to highlight regions with high spatial derivatives.

Non-maximum suppression: The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum (non-maximum suppression).

Hysteresis: The gradient array is now further reduced by hysteresis. Hysteresis is used to track along the remaining pixels that have not been suppressed. Hysteresis uses two thresholds and if the magnitude is below the first threshold, it is set to zero (made a non-edge). If the magnitude is above the high threshold, it is made an edge. And if the magnitude is between the two thresholds, then it is set to zero unless there is a path from this pixel to a pixel with a gradient above T2.

First step (Noise Removal)

In order to implement the canny edge detector algorithm, a series of steps must be followed. The first step is to filter out any noise in the original image before trying to locate and detect any edges. This is done so that the presence of noise will not introduce false edges. Gaussian filter is used in Canny algorithm because it can be computed using a simple mask. Once a suitable mask has been calculated, the Gaussian smoothing can be performed using standard convolution methods. A convolution mask is usually much smaller than the actual image. As a result, the mask is slid over the image, manipulating a square of pixels at a time. The Gaussian filter is shown in equation 9 below.

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right] \text{..... ....(9)}$$

The larger the width of the Gaussian mask, the lower is the detector's sensitivity to noise. The localization error in the detected edges also increases slightly as the Gaussian width is increased. A typical 5x5 Gaussian mask that can be used in noise removal is shown in figure 2.9 below.

$$\frac{1}{115}$$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Figure 2.13: Discreet approximation to Gaussian functions with $\sigma = 1.4$

Second step (finding gradients): The edges should be marked where the gradients of the image has large magnitudes.

An edge in an image may point in a variety of directions, so the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. The edge detection operator returns a value for the first derivative in the horizontal direction (Gx) and the vertical direction (Gy). From this the edge gradient and direction can be determined.

Canny edge detector finds the magnitude of the gradient and the direction of the individual pixels after filtering by using first-order differential operator. Using partial derivatives, the partial derivatives of the two directions of the chosen point (x,y) are:

$$G_x(i,j) = \frac{[I(i,j+1) - I(i,j) + I(i+1,j+1) - I(i+1,j)]}{2} \quad \text{.....(10)}$$

$$G_y(i,j) = \frac{[I(i,j) - I(i+1,j) + I(i,j+1) - I(i+1,j+1)]}{2} \quad \text{.....(11)}$$

The magnitude of the gradient is:

$$M(i,j) = \sqrt{G_x{}^2(i,j) + G_y{}^2(j,j)} \quad \text{.....(12)}$$

And its direction is:

$$\theta(i,j) = \tan^{-1}\left(\frac{G_x(x,y)}{G_y(x,y)}\right) \quad \text{.....(13)}$$

The edge direction angle is rounded to one of four angles representing vertical, horizontal and the two diagonals (0, 45, 90 and 135 degrees for example).

Third step: Non-maximum suppression.

Given estimates of the image gradients, a search is then carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction. So, for example, if the rounded gradient angle is zero degrees (i.e. the edge is in the north-south direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes in the west and east directions, if the rounded gradient angle is 90 degrees (i.e. the edge is in the east-west direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes in the north and south directions, if the rounded gradient angle is 135 degrees (i.e. the edge is in the north east-south west direction) the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes in the north west and south east directions, if the rounded gradient angle is 45 degrees (i.e. the edge is in the north west-south east direction)the point will be considered to be on the edge if its gradient magnitude is greater than the magnitudes in the north east and south west directions. From this stage referred to as

non-maximum suppression, a set of edge points, in the form of a binary image, is obtained. These are sometimes referred to as "thin edges".

Fourth step (Hysteresis thresholding)

Large intensity gradients are more likely to correspond to edges than small intensity gradients. It is in most cases impossible to specify a threshold at which a given intensity gradient switches from corresponding to an edge into not doing so. Therefore Canny uses thresholding with hysteresis.

Thresholding with hysteresis requires two thresholds – high and low. Making the assumption that important edges should be along continuous curves in the image allows us to follow a faint section of a given line and to discard a few noisy pixels that do not constitute a line but have produced large gradients. Therefore we begin by applying a high threshold. This mark out the edges we can be fairly sure are genuine. Starting from these, using the directional information derived earlier, edges can be traced through the image. While tracing an edge, the lower threshold is applied, allowing faint sections of edges to be traced as long as a starting point is found.

Once this process is complete we have a binary image where each pixel is marked as either an edge pixel or a non-edge pixel. From complementary output from the edge tracing step, the binary edge map obtained in this way can also be treated as a set of edge curves, which after further processing can be represented as polygons in the image domain.

2.13. Image segmentation

According to Gonzales and Woods (2002), segmentation refers to the operation of partitioning an image into component parts, or into separate objects. In Dwayne (2000), Image segmentation was defined as the process of dividing an image into regions or objects. It is the first step in the task of image analysis. Image processing displays images and alters them to make them look better, while image analysis tries to discover what is in the image. The basic idea of image

segmentation is to group individual pixels (dots in the image) together into regions if they are similar. Similar can mean they are the same intensity (shade of gray), form a texture, line up in a row, create a shape and so on.

Algorithms are available for segmentation and these algorithms are called segmentation algorithms. In Salem, Kalyankar and Khamitkar (2010), segmentation algorithms are based on one of two basic properties of intensity values discontinuity and similarity. First category is to partition an image based on abrupt changes in intensity, such as edges in an image. Second category is based on partitioning an image into regions that are similar according to predefined criteria. Histogram threshold falls under this category which is the focus of this project work. There are many techniques available for image segmentation, and the techniques vary in complexity, power, and area of application (Dwayne, 2000). Usually image segmentation is an initial and vital step in a series of processes aimed at overall image understanding. Applications of image segmentation include:

i. Identifying objects in a scene for object-based measurements such as size and shape
ii. Identifying objects in a moving scene for object-based video compression *(MPEG4)*
iii. Identifying objects which are at different distances from a sensor using depth measurements from a laser range finder enabling path planning for a mobile robot.

2.14 Segmentation techniques

Some of the techniques available for image segmentation will be reviewed next.

2.14.1 Thresholds

Threshold is one of the widely methods used for image segmentation. It is useful in discriminating foreground from the background. By selecting an adequate threshold value T, the gray level image can be converted to binary image. The binary image should contain all of the essential information about the position and shape of the objects of interest (foreground). The

advantage of obtaining first a binary image is that it reduces the complexity of the data and simplifies the process of recognition and classification. The most common way to convert a gray-level image to a binary image is to select a single threshold value (T). Then all the gray level values below this T will be classified as black (0), and those above T will be white (1). The segmentation problem becomes one of selecting the proper value for the threshold T. A frequent method used to select T is by analyzing the histograms of the type of images that want to be segmented (Salem, Kalyankar and Khamitkar, 2010).

$$g(x,y)=\begin{cases}1 \; if \; f(x,y)>T \\ 0 \; if \; f(x,y)\le T\end{cases} \quad \text{.....(14)}$$

When T depends only on f(x,y) it refers to as global threshold, if T depends on f(x,y) and p(x,y) then it is called as local threshold and if T depends on x ,y along with p(x ,y) and f(x ,y) then it is called as adaptive threshold. In the histogram of an image, the highest peak represents the background while the smaller peak represents the object. This is shown in figure 2.13 below.

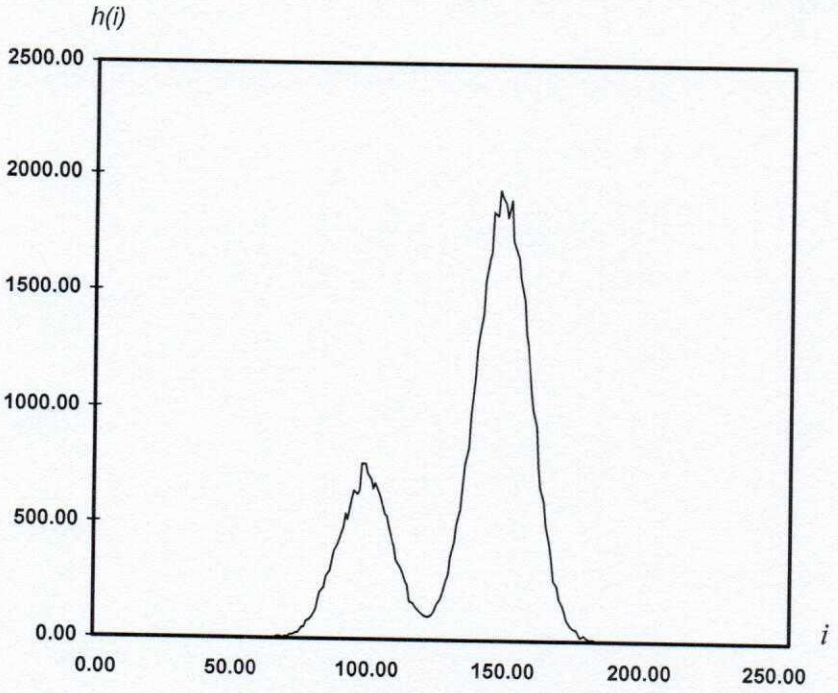


Figure 2.13: Graylevel thresholding

Graylevel thresholding algorithm can be defined as follows:

(i) If the Graylevel of pixels **p <=T** then pixel p is an object

(ii) Else, pixel p is a background pixel.

2.14.2 Graylevel Clustering

Clustering is a process of organizing the objects into groups based on its attributes. An image can be grouped based on keyword or its contents. Keyword describes the similar features of an image, whereas content refers to shape, texture etc. Both supervised and unsupervised clustering techniques are used in image segmentation (Gurjeet and Rajneet, 2013). Clustering tries to separate an histogram into two groups defined by two clusters C1 and C2. In this case, graylevels are classified according to the nearest cluster center. An idealized object/background histogram is shown in figure 2.14 below

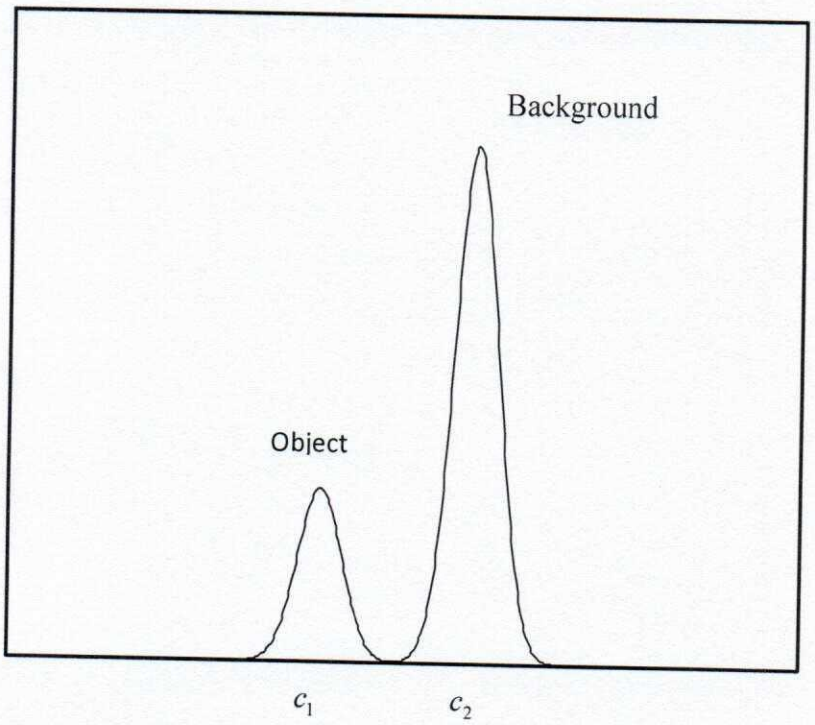


Figure 2.14: Object/background clusters

A nearest neighbour clustering algorithm allows us to perform a Graylevel segmentation using clustering. A simple case of a more general and widely used K-means clustering uses simple iterative algorithm which has known convergence properties.

Given a set of graylevels:

$$\{g(1),\ g(2).....\ g(N)\}$$

We can partition this set into two groups:

$$\{g_1(1), g_1(2).....g_1(N_1)\}$$

$$\{g_2(1), g_2(2).....g_2(N_2)\}$$

Then we can compute the local means of each group as

$$c_1 = \frac{1}{N_1}\sum_{i=1}^{N_1} g_1(i) \qquad \text{.....(15)}$$

$$c_2 = \frac{1}{N_2}\sum_{i=1}^{N_2} g_2(i) \qquad \text{.....(16)}$$

We then Re-define the new groupings

$$|g_1(k) - c_1| < |g_1(k) - c_2| \quad k = 1..N_1 \text{.....(17)}$$

$$|g_2(k) - c_2| < |g_2(k) - c_1| \quad k = 1..N_2 \text{.....(18)}$$

In other words all grey levels in set 1 are nearer to cluster centre c_1 and all grey levels in set 2 are nearer to cluster centre c_2

We now initialize the label of each pixel randomly

Repeat

c_1= mean of pixels assigned to object label

c_2= mean of pixels assigned to background label

Compute partition $\{g_1(1), g_1(2).....g_1(N_1)\}$

Compute partition $\{g_2(1), g_2(2).....g_2(N_2)\}$

Until none pixel labelling changes

We can show that the algorithm is guaranteed to converge and also that it converges to a sensible result. To do this we define a 'cost function' at iteration r

$$E^{(r)} = \frac{1}{N_1}\sum_{i=1}^{N_1}\left(g_1^{(r)}(i) - c_1^{(r-1)}\right)^2 + \frac{1}{N_2}\sum_{i=1}^{N_2}\left(g_2^{(r)}(i) - c_2^{(r-1)}\right)^2 \text{..... ..... (19)}$$

$E^{(r)}>0$

We now update the cluster centres and finally update the cost function as

$$c_1^{(r)} = \frac{1}{N_1}\sum_{i=1}^{N_1} g_1^{(r)}(i) \text{..... ....} .(20)$$

$$c_2^{(r)} = \frac{1}{N_2}\sum_{i=1}^{N_1} g_2^{(r)}(i) \text{..... ....} .(21)$$

$$E_1^{(r)} = \frac{1}{N_1}\sum_{i=1}^{N_1}\left(g_1^{(r)}(i) - c_1^{(r)}\right)^2 + \frac{1}{N_2}\sum_{i=1}^{N_2}\left(g_2^{(r)}(i) - c_2^{(r)}\right)^2 \text{..... ....} (22)$$

E_1 is simply the sum of the variances within each cluster which is minimised at convergence and this gives sensible results for well separated clusters. This gives similar performance to thresholding as shown below in figure

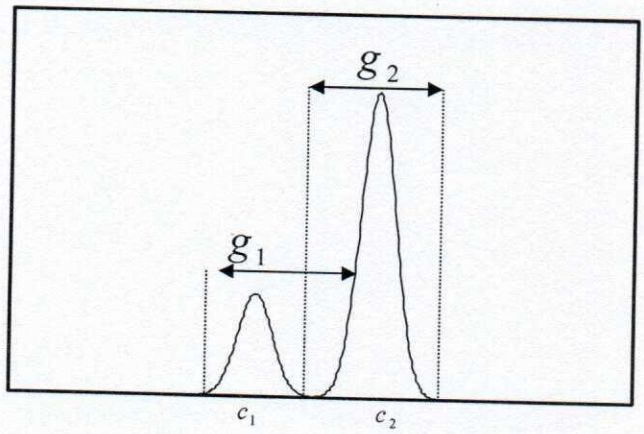
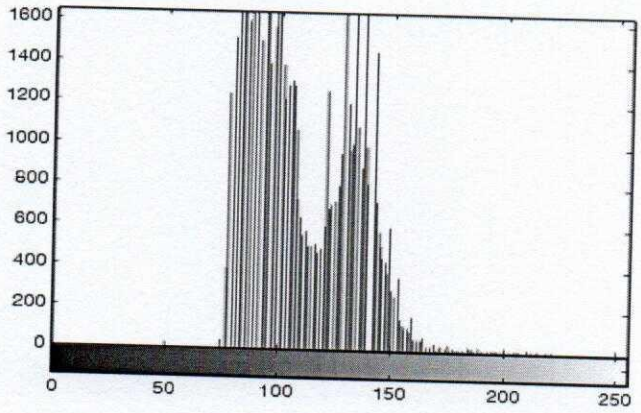


Figure 2.15: Graylevel clustering

2.14.3 Histogram-based segmentation

Histogram-based segmentation uses the histogram to select the gray levels for grouping pixels into regions. In a simple image there are two entities: the background and the object. The background is generally one gray level and occupies most of the image. Therefore, its gray level is a large peak in the histogram. The object or subject of the image is another gray level, and its gray level is another, smaller peak in the histogram (Gonzales and Woods, 2002).

Figure 2.16 (a) Original image



(b) Histogram of the image.

When the gray levels in an image are grouped together at the dark end of the histogram, it indicates a poor contrast. When the gray levels are evenly distributed along the histogram, this indicates a very good contrast image.

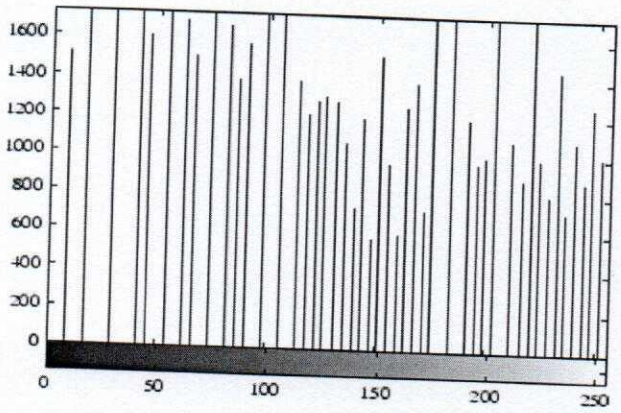


Figure 2.17: Histogram of good contrast image.

Histogram equalization

Images that are poorly scanned will have their gray levels concentrated in the dark regions of their histogram and therefore have poor contrast. Objects in such images cannot be seen clearly. Therefore to extract those objects, their contrasts have to be enhanced. This can be done by evenly distributing their gray levels along their histogram just like the histogram in figure 2.17 above. The process of enhancing image brightness or improving its contrast is called histogram equalization. Histogram equalization ensures that objects in an image are bright and clearly seen because their gray values are no longer concentrated in the dark portion of the histogram. The focus of this project work is to use histogram equalization as a tool for segmenting image. Histogram equalization uses the results of one histogram to transform the original image into an image that will have an equalized histogram.

The histogram equalization algorithm may make more sense by outlining the derivation of the underlying mathematical transform. Equation (23) represents the equalization operation, where c

is an image with a poor histogram. The as yet unknown function f transforms the image c into an image b with a flat histogram.

$$b(x,y) = f[c(x,y)]..... ....(23)$$

The equation (24) below shows the probability-density function of a pixel value a. p1(a) is the probability of finding a pixel with the value a in the image. Area1 is the area or number of pixels in the image and H1(a) is the histogram of the image.

$$p_1(a) = \frac{1}{Area_1} H_1(a).....(24)$$

For example if a = 100, area1 = 10,000. H1(100) =10. Then p1(100)=10/10,000 = 0.001.

Equation (25) shows the cumulative-density function (cdf) for the pixel value a. The cdf is the sum of all the probability density function up to the value a.

$$p_1(a) = \frac{1}{Area_1} \sum_{i=0}^{a} H_1(a).....(25)$$

For example,

p1(10) = 1/10,000*[H(0) + H(1) + … + H(10)]

Equation (26) shows the form of the desired histogram equalization function f(a). $H_c(a)$ is the histogram of the original image c (the image with the poor histogram). D_m is the number of gray levels in the new image b. D_m = 1= p(a) for all pixel values a in the image b. Note that the image b has a flat histogram H(0) = H(1) = H(2) = ::: because the probability of each pixel value is now equal - they all occur in the same number of times. So f(a) simply takes the probability density function for the values in image b and multiplies this by the cumulative density function

of the values in image c. It is important to realize that histogram equalization reduces the number of gray levels in the image.

This seems to be a loss, but it is not according to Gonzales and Woods (2002).

$$f(a) = D_m \frac{1}{Area} \sum_{i=0}^{a} H_c(a) \quad \text{.....(26)}$$

CHAPTER THREE

MATERIALS AND METHOD

3.1 Introduction

This chapter deals with how the project work is carried out. It shows the detailed description of the analysis and the available processing tools in the application software on which the work is being done. It also gives a brief introduction to the system elements, its applications, working and the structure of the proposed system. It gives the problem definition and the proposed system solution and about the implementation of various tools of application software for simulation and the testing part of the project. That is the methodology. This includes the materials and method. The material used for the project is the software known as MATLAB (an acronym for Mathematics Laboratory). This is a high level language that can be used to develop software or for simulation. This software is used to implement the segmentation of input image using histogram equalization. We will now look at the language features of MATLAB.

3.2 Language Features for Matlab

Matlab language has the following characteristics:

(i) **Programming efficiency**

It is a scientific and engineering calculation for high-level language that allows the form of mathematical programming language. Fortran and C and other languages written in the formula closer to our way of thinking, like using Matlab programming paper in calculus arrange on a formula and solve problems. Therefore, Matlab language, popularly known as the calculus can also be paper-based science algorithm language because it is easier to write. So the programming and high efficiency, everyone can easy to learn.

(i) **Easy to use**

Matlab language is an interpreted language (in no special tools are compiled before), it is flexible. That means of its rich debugger, debugging speed, less time to learn. Matlab language compare with other languages, has solved these problems, edit, compile, link and execute integration. It can be flexible in the same operation on the screen quickly ruled out entering the program in writing. And Matlab also have grammatical errors as well as semantic errors, thereby speeding up the user to write, modify and debug programs faster. If people used Matlab, people will know Matlab's advantage.

(ii) **Expansion capability**

High version of the Matlab language has rich library functions, when carrying out complex mathematical operations can be called directly. And the Matlab library functions in the form on file with the same user, so users can file as a library of Matlab functions to call. Thus, the user can easily build own needs and the expansion of the new library functions. Using of Matlab in order to improve efficiency and expand its capabilities.

(iii) **Statement is simple, rich in content**

The most important ingredient of Matlab language is the most basic function of a varying number of input variables and different number of output variables, representing different meaning (a bit like object-oriented polymorphism). This is not only makes the Matlab library function more feature-rich, while greatly reducing the need for disk space, making the Matlab of M-file written in a simple, short and efficient.

(iv) **Efficient and convenient matrix and array operations**

Matlab language provides for matrix arithmetic operators, relational operators, logical operators, conditional operators and assignment operator. But most of these operators can be copied without any change in operations between the array, such as arithmetic operators increase the "·"

can be used for operations between arrays. It need not define the dimension of the array, and gives the matrix function. It is much simple, efficient and convenient.

(v) **Convenient graphics**

Matlab graphics is very convenient. It has a range of drawing functions. In addition, call the drawing function to adjust the color argument can be drawn the same point, line, double line or multiple lines. The design for the sake of this project is a common programming language that fall. In short, Matlab language design can be said to represent the current high-level computer language development.

3.3 Introduction to MATLAB

When it comes to image processing using Matlab there are many things to keep in mind like using the right format, loading an image, reading an image, how to display an image, keeping the saved data in different types, writing an image, converting different type of image formats. Here we discuss some of the functions (commands) used for this type of operations. Matlab has got 'Image processing toolbox' which is necessary when working with images in order to use the commands in the toolbox. This tool box is installed in the full version of Matlab software and to check whether it is installed and available, type "ver" in the Matlab prompt. When that is typed we can see all the tool boxes that are installed and available to use on your system. You can use Matlab's help browser option if you have any further problems and references. Matlab has got many easy commands to process many types of functions very fast and easily. You can have access to online manual for image processing toolbox and any kind of tool boxes which you can access through Matlab help browser.

3.4 Digital image in MATLAB

An image is a 2D array of values representing light intensity. in case of image processing the term image represents or refers to a digital image. Image is a function of light intensity. For

example an image is represented as the function f(x,y) where f is the brightness of the point(x ,y), and x, y represents the spatial coordinates of a picture element known as pixels. By convention, the spatial reference of the pixel with the coordinates (0,0) is located at the top- left corner of the image as shown in the figure 3.1..

3.5 Images in Matlab

There are many types of image formats, most of the formats are accepted by Matlab. The image formats that are supported by Matlab are:

(i) JPEG

(ii) PCX

(iii)TIFF

(iv)BMP

(v) HDF

The mostly used format for an image is JPEG- images and this is the main format whose images are found on internet. It is the most widely used compression standards for images. When we store an image we can see from the suffix in what format it is stored. For example an image is stored as "baboon.bmp" is stored in the bmp format and we can see later on how to load an image of this format into Matlab. Also, If an image is stored as "cat.jpg" then we will see that it is stored in jpg format so that we will be able to load an image into Matlab of this format. The image that will be used in this project work will be in Jpeg format for the reason given above.

3.6 Working formats in Matlab

If an image is stored as a JPEG-image format on your disc or drive we must first read into Matlab. Whenever we started working with an image or to perform any operation on image like performing a wavelet transform or Hough transform on the image, we need to convert the image into a different format. Here some of the formats are discussed.

3.6.1 Binary image

This image format also stores an image as a matrix but can only colour a pixel black or white (and nothing in between). It assigns a 0 for black and a 1 for white.

3.6.2 Intensity image (Gray scale image)

Intensity image is almost equivalent to "gray scale image" and we mostly use this type of image. It is a matrix in which every element will have a value relative to how intensively pixel is in the corresponding position must be coloured. In two ways the number can be represented which will gives the details of the pixel intensity. The first one is data type or it is called as double class which is used to assigning a floating number between zero (0) and one (1) to every pixel. Zero means black colour and white implies to 1. Second class is unit8 which will assign a number or integer from 0 – 255 to represent the intensity of the pixel. Here, 0 represent black while 255 represent white. The same way 0 implies to black and 1 to white. This class requires nearly 1 by 8th of the storage when compared with the first class. But most of the mathematical functions can be applied in the first type of class that is double class. In this project the work is mostly done with gray scale images.

3.6.3 Indexed Image

This represents colour images in a practical way. Indexed image stores image as two matrices. In the first matrix it has the same size as image and one number for each pixel and in the second matrix its size may be different from image and it is called as colour map. The numbers in the first matrix is an instruction of what number to use in the colour map matrix.

3.6.4 RGB image (Red, Green, Blue)

This format is for colour images. In this type of format an image is shown with 3 matrix with the same sizes that match format of the image. In this every matrix implies to 1 of colours in red, green or blue. The matrix gives instructions to pixels about the certain amount of these colours

that should be used. This type of colour image was discussed in the literature review in this project work.

3.6.5 Multi frame image

Multi frame format is good for sequence of images. There are some areas where we may need to work with the image sequences. This type of image is often used in medical and biological imaging systems and is mostly in 3-D format because time is involved in the capture of the image.

However, these image types can be converted from one form to the other depending on the context in which they are used during image processing. When working with images in Matlab there will be a need to change the image formats. Below are examples of some conversion functions that could be found in Matlab. Since every function needs an argument or arguments to perform a task, every Matlab function need at least one argument, enclosed in parenthesis, to be supplied. The name of the image we want to use in this case, serves as the argument to the function. Some of the functions are:

(i) Graythresh( ) –This is used for global image thresholding

(ii) im2bw( ). This function converts image to binary image based on threshold

(iii) Ind2gray( ) – This function converts indexed image to gray scale image

(iv) dither( ) - converts intensity/indexed/RGB format to binary format

(v) gray2ind( ) - converts intensity format to indexed format

(vi) ind2rgb( ) -converts indexed format to RGB format

(vii) mat2gray( ) -converts a regular matrix to intensity format by scaling

(viii) rgb2gray( ) -converts RGB format to intensity format

(ix) rgb2ind( ) - converts RGB format to indexed format

All the commands mentioned above needs image processing toolbox for them to perform their operations.

3.6.6 Reading & writing image files

When an image is considered which we want to work with, it is usually in a file format like when we download an image from the web it is usually stored as a JPEG file. When the image processing has completed, we may want to write it back to a JPEG-file so that we can, like posting the image on the internet. To do these things there are commands meant to perform these operations. They are **imread** and **imwrite**. These commands also require image processing toolbox.

(a) **imread():** This command is used to read an image. The name of the file to be read has to be kept within the parenthesis. The name of the file has to be in single quotes ' '.

(b) **imwrite(,):** This command is used to write an image to a file. The name of the image we have worked with has to be typed within the parenthesis as the first argument. As a second argument within the parenthesis type the name of the file and format that we want to write the image to. The name of the file has to be kept within the single quotes ' '. Semi-colon should be used after these commands, otherwise lots of number scrolls on the screen.

3.6.7 Displaying image in Matlab

Here we discuss about couple of basic Matlab commands used for displaying an image. These commands do not require any tool box.

(i) **imagesc(x):** Display an image represented as the matrix x.

(ii) **brighten (s):** it adjusts the brightness. S is a parameter such that -1<s<0 gives a darker image, 0<s<1 gives a brighter image.

(iii) **colormap(gray):** it changes the colours to gray. There will be times when image is not displayed in gray scale though after conversion to grayscale. Then colormap(gray) command can

be used to force Matlab to make that image to be displayed in gray scale. When Matlab is used with image tool processing box then it is recommended to use the imshow command to display the image.

Displaying image on matrix form: There are some commands to display an image that is given on matrix form. It requires image processing tool box.

(i) **imshow(x):** Display an image represented as the matrix x.

(ii) **zoom on:** it is used to zoom in using the left and right mouse button

(iii)**zoom off:** it is used to turn off the zoom function.

3.7 MATLAB WORKSPACE

This project work was implemented with MATLAB 7.7.0 (2008b). After installation, to load MATLAB, click on start button in the Microsoft Windows Operating System and browse to MATLAB 2008b. When found, click on it and the program loads and displays the following window as shown in figure 3.2..

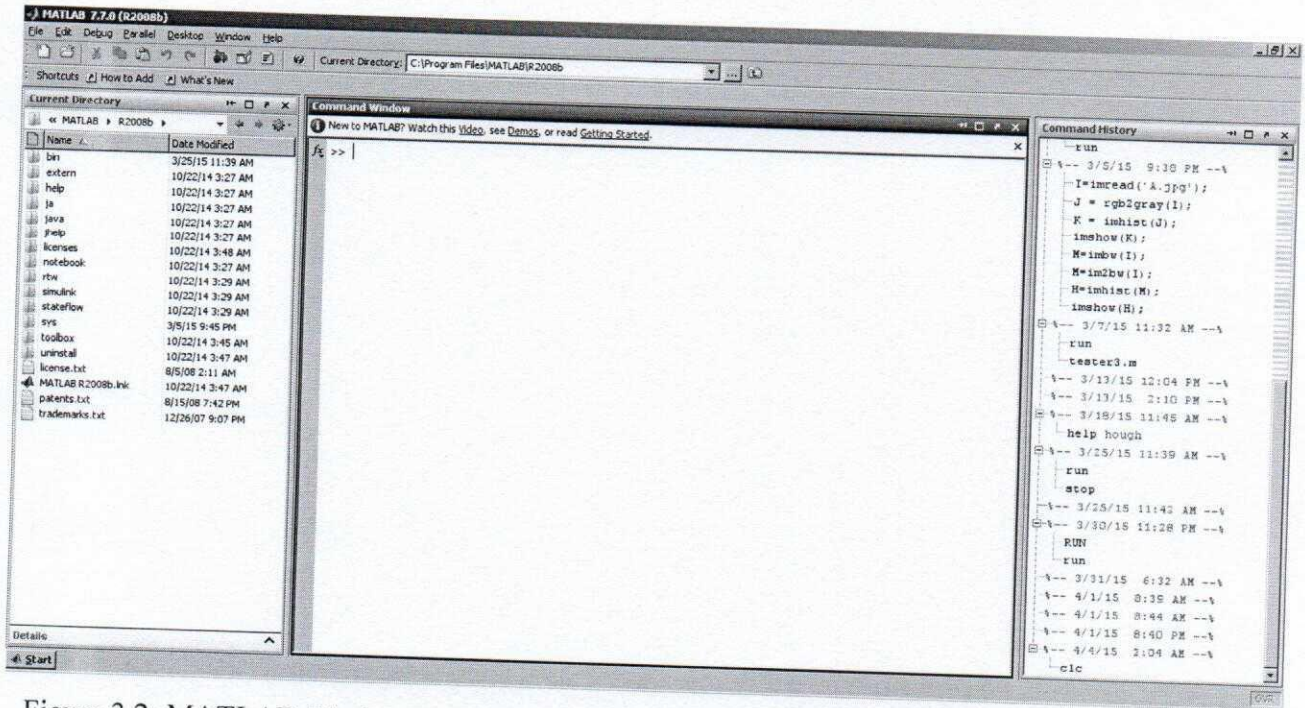


Figure 3.2: MATLAB Workspace

The workspace is divided into three windows discussed below.

Current Directory: The first on the left is the current directory window that displays the list of folders that are in the current directory where we are working. The directories as well as the date they are created are listed as well. The current directory can be changed by selecting the path to the folder we are working with.

Command Window: The second window at the middle is the command window where commands that processes our input image are entered and executed. Each command is terminated with a semicolon in MATLAB. After entering the command, to execute it we simply press the ENTER KEY on the keyboard. The command window also list all the command entered as well as error messages listing the errors our code displays for us. These messages can be cleared by entering the command "clc" and pressing enter key.

Command History: this window portion contains the list of all the commands that have been executed recently. They can however be cleared by right-clicking inside the window and select the command "clear history".

3.8 Getting help with MATLAB

If you are a Matlab beginner you can click on **Getting Started** for general help with how Matlab works or click on **User's Guide | Programming fundamentals | Syntax Basics** for basic information such as how variables are created and initialized in Matlab. An excellent description of Matlab expressions can be found in **Getting Started | Matrices and Arrays |Expressions**. It points out the fact that Matlab stands for "Matrix Laboratory". **Whenever possible use a matrix expression** instead of a for loop to make matrix calculations. These expressions will execute much faster than nested for loops, because Matlab is optimized for manipulating matrices.

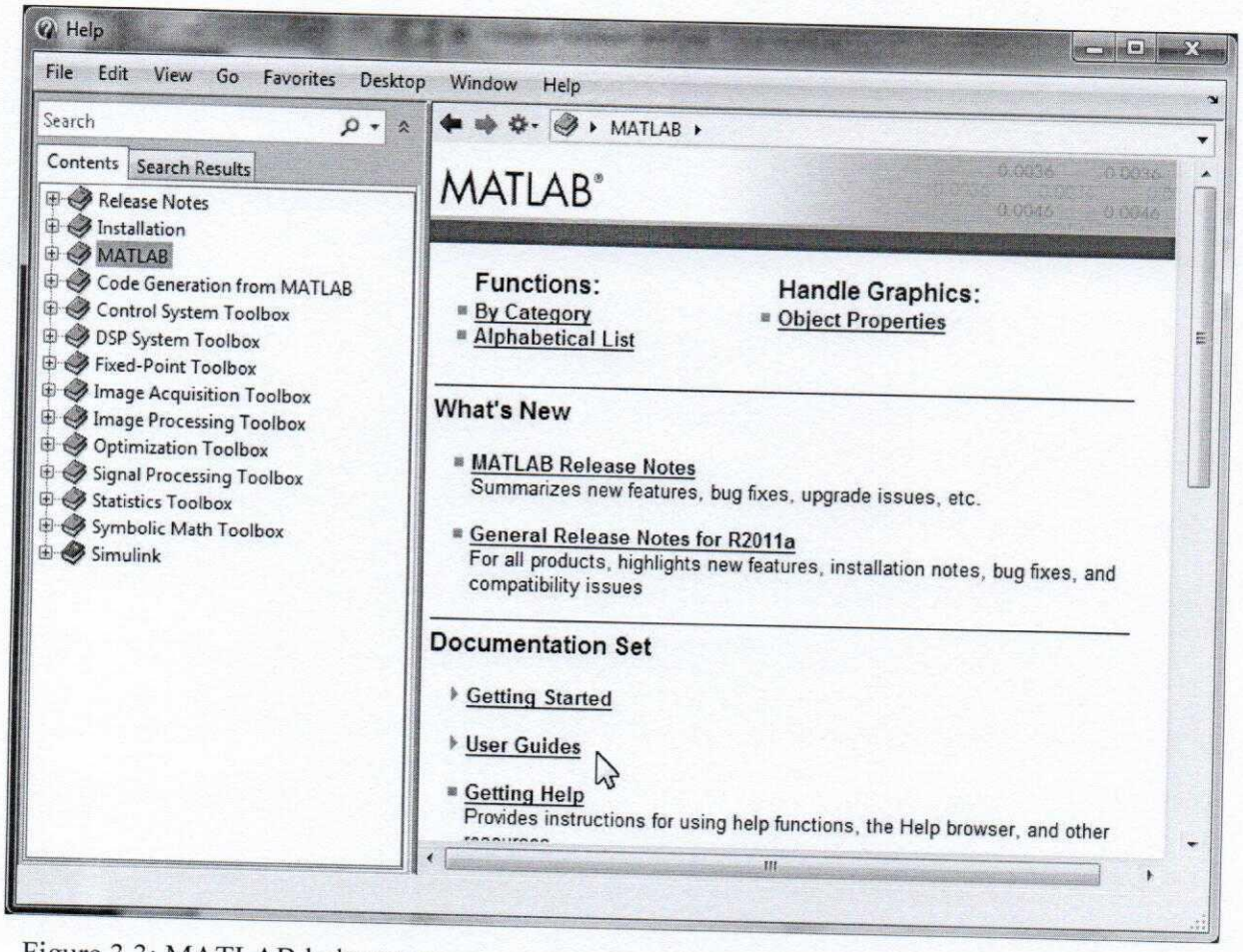


Figure 3.3: MATLAB help screen.

3.9 Creating writing and running programs in MATLAB (m-files)

MATLAB files are called m-files. We can run commands in the **Command Window** to try out how they work. For automatically running commands, create an m-file. This is the "source code" for MATLAB programs. m-files are interpreted programs, called scripts, and are not compiled before running. Remember: Matlab is a "Computational Program", not our usual programming language. m-files can be created in two ways: type edit in the **Command window** or click on the Actions icon in the **Current Folder** window and select **New File | Script**: This is shown in figure 3.4.

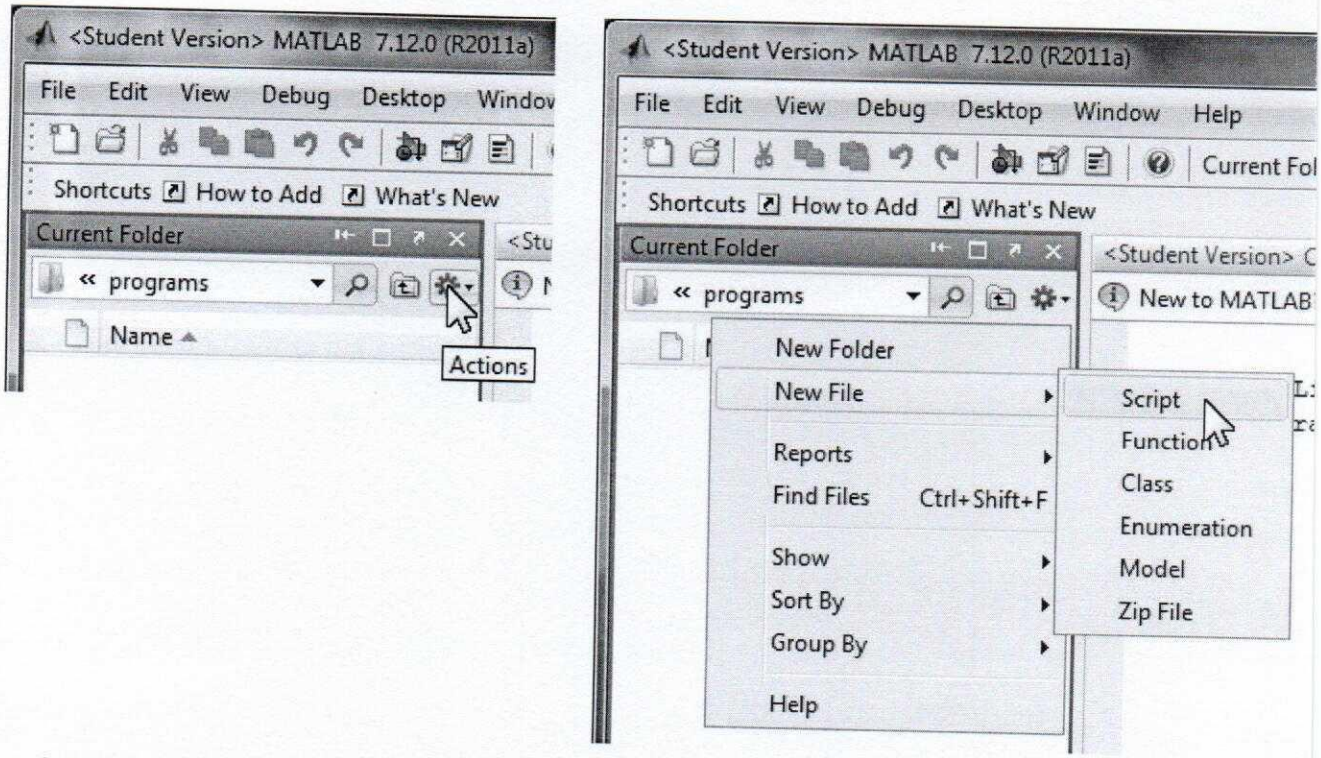


Figure 3.4: Creating programs in MATLAB

The **Editor - Untitled** window will pop up (when you save it you can give it a name): this is as shown in figure 3.5.

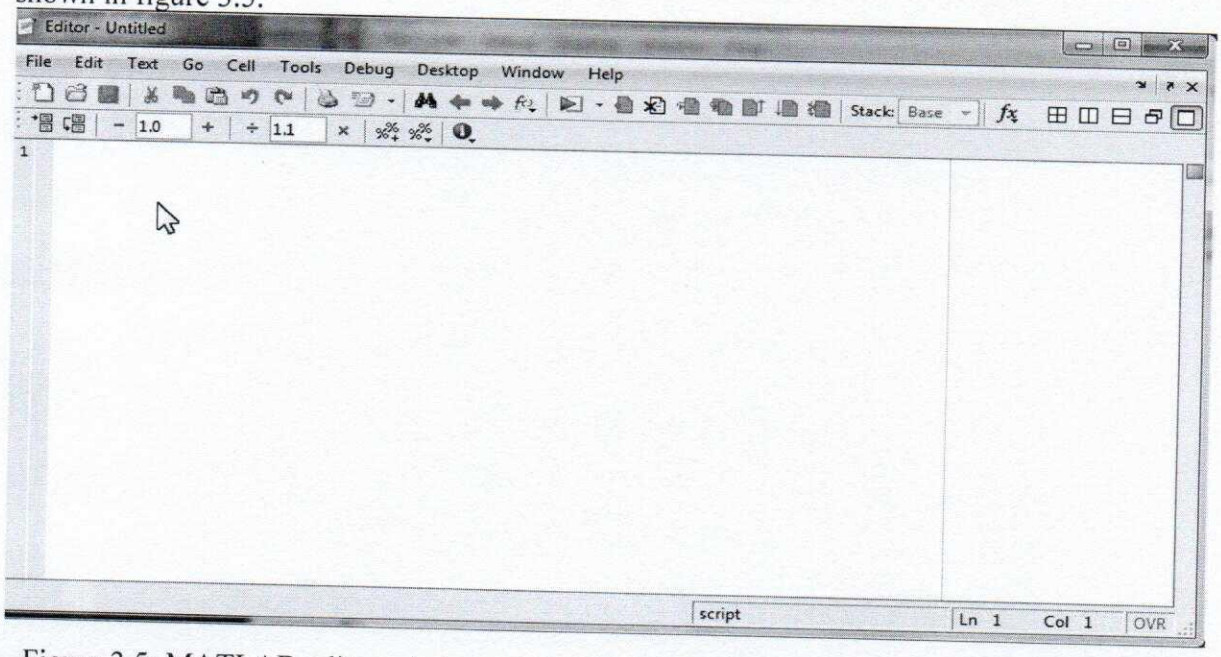


Figure 3.5: MATLAB editor window.

3.10 Reading and Writing images (matrices of pixel data)

Type images in Search box and click on first result labeled as a basic MATLAB feature available without needing the Image Processing Toolbox. The screenshot below shows as in figure 3.6.

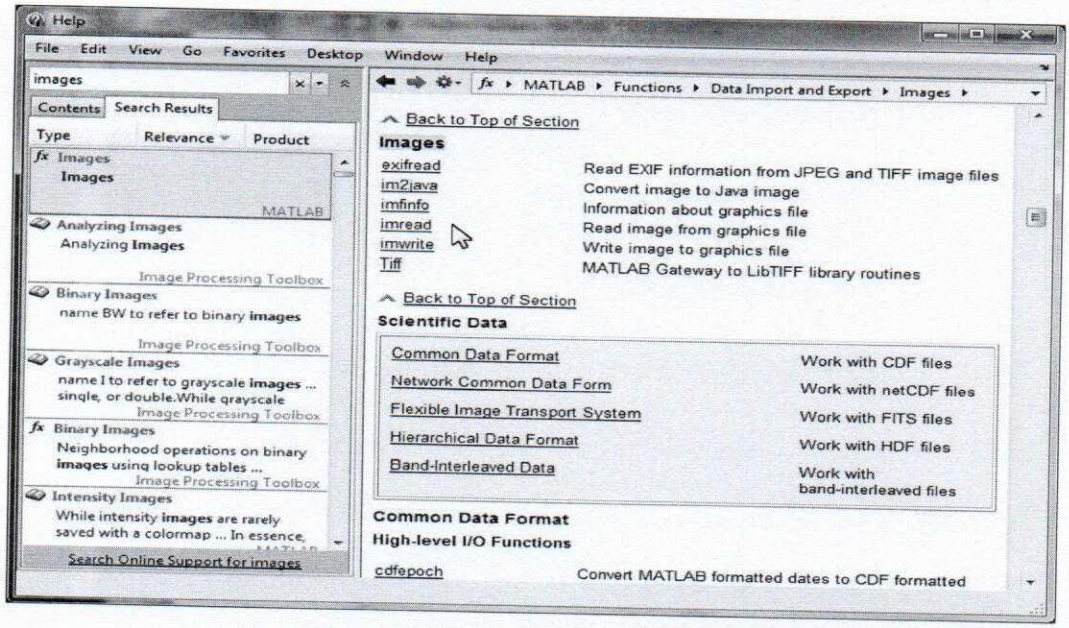


Figure 3.6: Reading and writing images

Reading images and displaying them (note the use of the semicolon to suppress command window echo). These are 3D matrices where each pixel is a 3-element vector:

```
1 clear all; close all; clc;
2 %load an image and display it
3 rgb_img = imread('Photo_062011_002.jpg');
4 image(rgb_img);
```

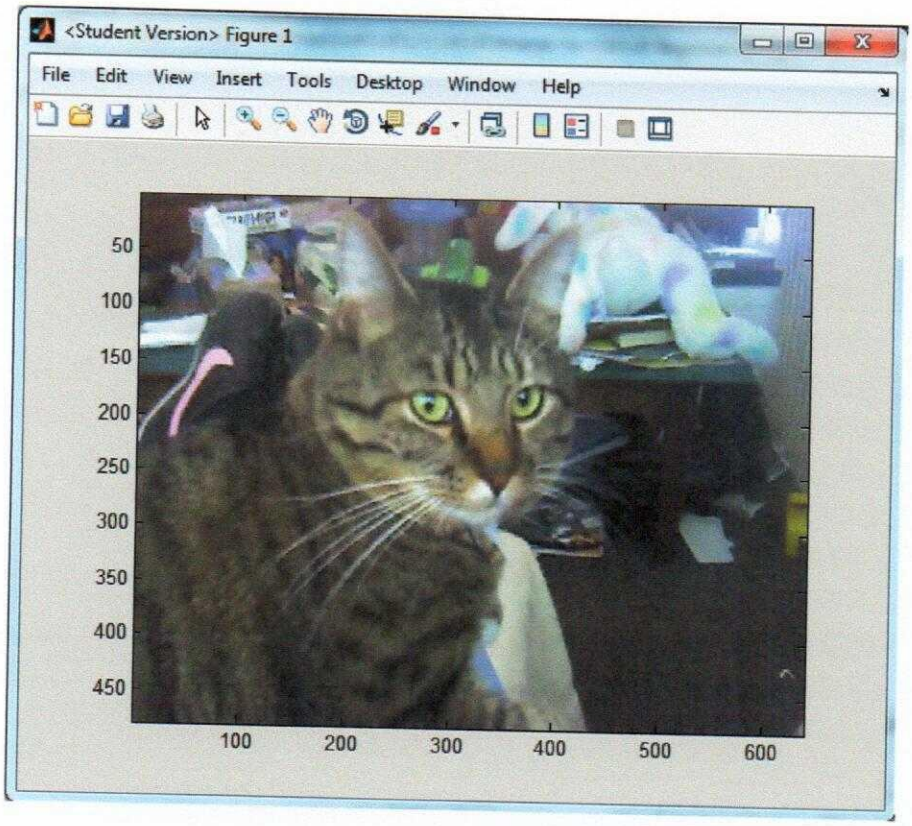


Figure 3.7: Displaying image

3.11 Convert color to gray scale

Find help: User's Guide | Graphics | Displaying Bit-mapped Images | Working with 8-Bit and 16-Bit Images | Converting an 8-Bit RGB Image to Grayscale

Example:

```
clear all; close all; clc;
%load an image and display it in figure 1
rgb_img = imread('Photo_062011_002.jpg');
image(rgb_img);
%fit plot box tightly around the image data
axis image;
%Change image to grayscale 2D matrix
I = .2989*rgb_img(:,:,1)...
+.5870*rgb_img(:,:,2)...
+.1140*rgb_img(:,:,3);
%display grayscaled image in figure 2 with gray(256) colormap
figure; colormap(gray(256)); image(I);
axis image;
```

After executing the code above, the result is shown in figure 3.8.

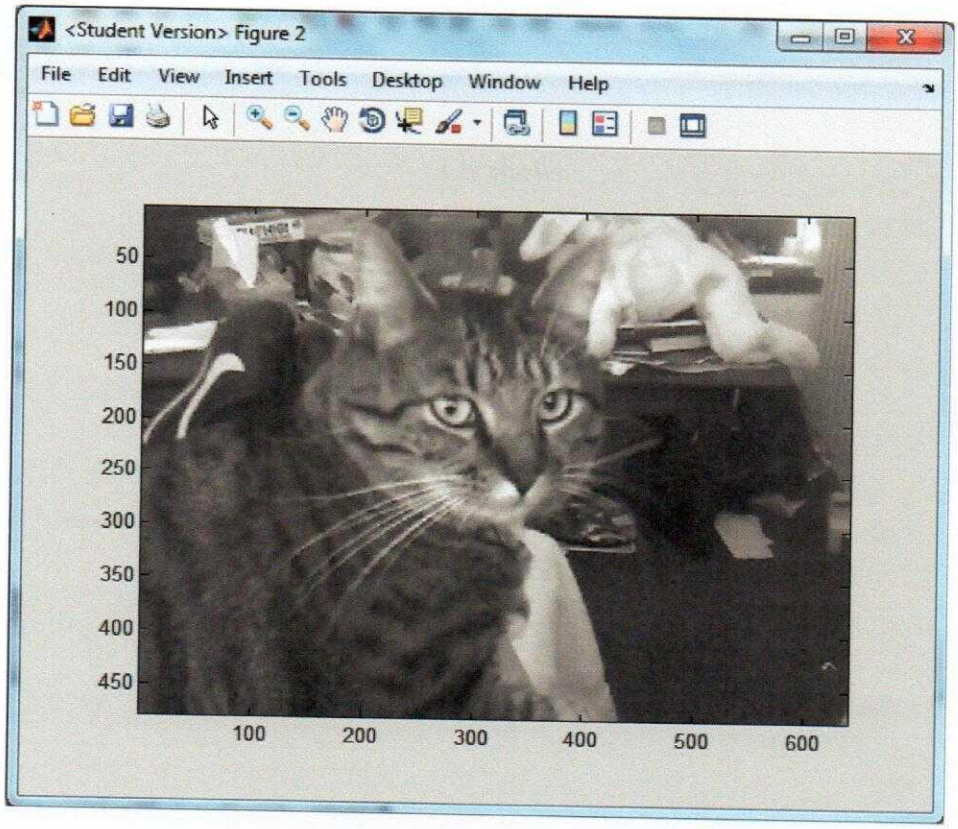


Figure 3.8: Display of the grayscale image.

3.12 Display images (multiple)

Sometimes need arises when we need to display multiple images in a single output image window. This can be done by the program below. We can use figure, image, and subplot commands.

```
clear all; close all; clc;
%load an image and display it in first row, 1st column, figure 1
im1 = imread('Photo_062011_002.jpg');
subplot(2,2,1);image(im1);
%fit plot box tightly around the image data
axis image;
%Change image to grayscale 2D image
I = .2989*im1(:,:,1)...
+.5870*im1(:,:,2)...
+.1140*im1(:,:,3);
%display grayscaled image in 1st row, 2nd column, figure 1
```

```
subplot(2,2,2); colormap(gray(256)); image(I);
axis image;
%load another image and display it in second row, 1st column figure 1
im2 = imread('IMG_1766.jpg');
subplot(2,2,3); image(im2);
axis image;
%load 3rd color image and display it in second row, 2nd column figure 1
im3 = imread('IMG_1768.jpg');
subplot(2,2,4); image(im3);
axis image;
```

The output is as shown in figure 3.9.

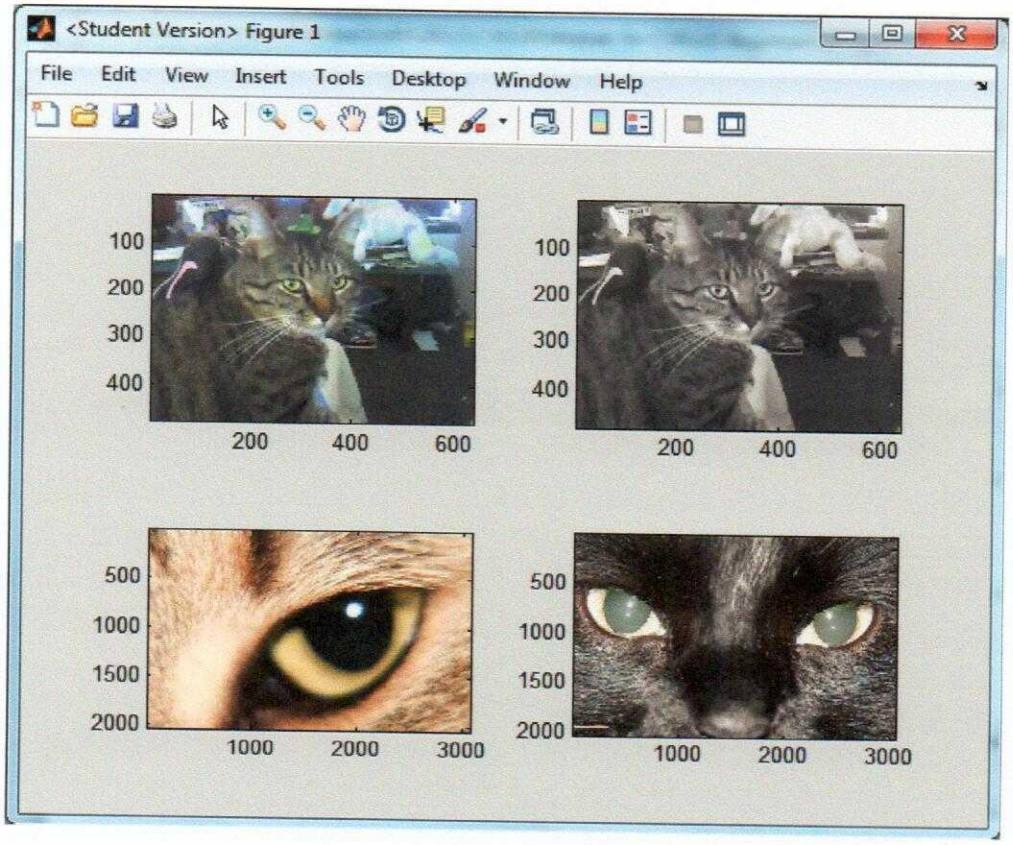


Figure 3.9: writing multiple images

3.13 Program Design

This project is designed using the top-down approach to programming. The executing of the program is from top to bottom in a straightforward approach.

3.14 Problem definition

Images that are poorly scanned will have their gray levels concentrated in the dark regions of their histogram and therefore have poor contrast. Objects in such images cannot be seen clearly. Therefore to extract those objects, their contrasts have to be enhanced. This can be done by evenly distributing their gray levels along their histogram before binarization of the image. This is done to make the task of image segmentation easier. Objects that are not distinct in images are difficult to segment and machine learning algorithms perform poorly on them. The process of enhancing image brightness or improving its contrast is called histogram equalization. Histogram equalization ensures that objects in an image are bright and clearly seen because their gray values are no longer concentrated in the dark portion of the histogram.

3.15 Structure of the proposed solution

The proposed solution to the above problem is outlined in the following stages.

3.15.1 Image Acquisition

The images used for this project work was acquired with the aid of a digital camera of a known resolution. The images are stored as a jpeg image format. MATLAB can work with images stored in any of the formats discussed in section 3.5 of this project work.

3.15.2 Program Algorithm

An algorithm is the basic steps that are followed which leads to the solution of a problem. The problem here is to segment an image using the method of histogram equalization. The algorithm used for implementing histogram equalization in this project work is listed below. The algorithm contains four basic steps.

Algorithm

Step 1. For every pixel in the image, get gray value in variable *i*. $hist[i] = hist[i] + 1$ when *i=0 to L-1* for a *L* level image.

Step 2. From the histogram array, get gray cumulative frequency of histogram

$$hist_{cf}[i] = hist_{cf}[i-1] + hist[i]$$

Step 3. Generate the equalized histogram as:

$$eqhist[i] = \left\lfloor \frac{[(L * hist_{cf}[i]) - N^2]}{N^2} \right\rfloor$$

Where, L is the number of gray levels present in the image, N^2 is number of pixel in NxN image, $\lfloor x \rfloor$ is truncation of x to the nearest integer.

Step 4. Replace the gray value *I,* by *eqhist[i]* for each *i.* the *eqhist* contains the new mapping of gray values.

3.15.3 Program flowchart

The program flowchart is a pictorial representation of the basic steps taken to solve a problem. Flowcharting is a graphical way of depicting a problem in terms of its inputs, outputs, and processes. The flowchart for the implementation of histogram equalization is shown in figure 3.10.

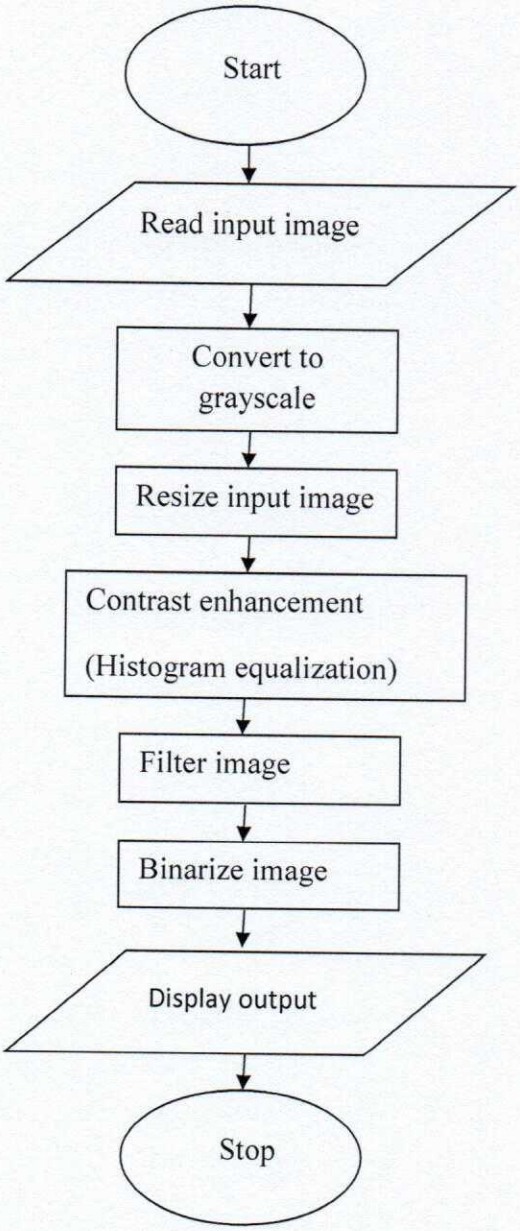


Figure 3.10: System flowchart

CHAPTER FOUR

SYSTEM IMPLEMENTATION

4.1 Introduction

The programming language used for the implementation of the project is MATLAB. The program was developed and created using MATLAB m-files. An m-file named *histogram.m* was created. The file was executed in the MATLAB's workspace and the result generated and presented as shown below.

4.2 Results and Discussion

The inputs to the system are two noisy images with poor contrast. The images are named "Tolu1" and "Tolu2". After preprocessing the image, a better image suitable for processing and segmentation was derived. The input, image as well as the corresponding output images, after histogram equalization are respectively shown in figure 4.1. Their image histograms are as well displays to show how the gray levels have been spread over the entire image to enhance it.

The histogram equalized image sets are shown first in figure 4.1 and 4.2.

4.3 Histogram equalization on image results

Before and after performing histogram equalization on the input images of Tolu1 and Tolu2 respectively, the results are shown in figure 4.3 and figure 4.4 respectively. Here the poor contrast of the input images has been enhanced and he edges highlighted very well. This is a necessary step to segmenting an image.

Figure 4.1a: Original Image

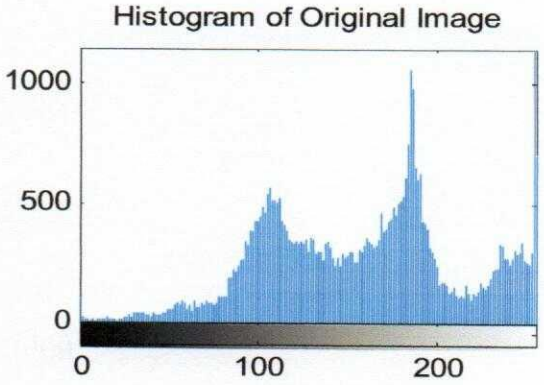


Figure 4.1b: Histogram of Original Image

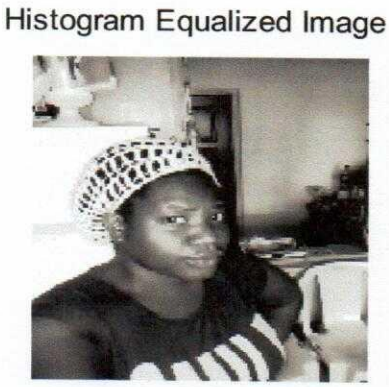


Figure 4.2a: Image after histogram equalization

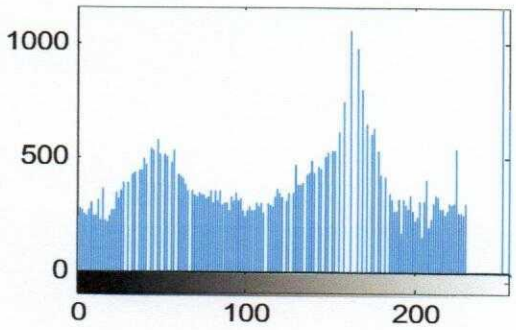


Figure 4.2b: Histogram of Histogram-equalized image

Figure 4.3a: Original Image

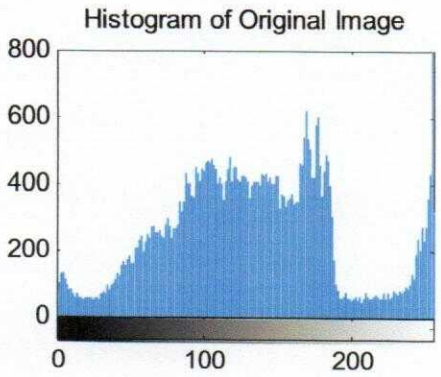


Figure 4.3b: Histogram of original image

Figure 4.4a: Image after histogram equalization

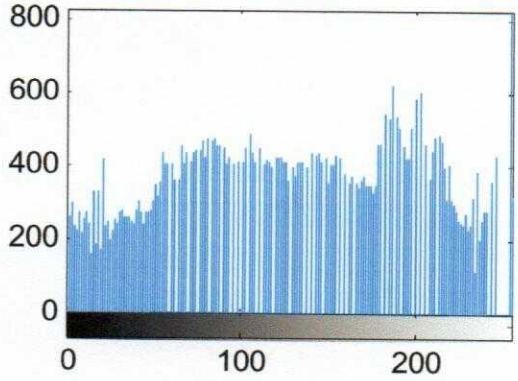


Figure 4.4b: Histogram of Histogram-equalized image

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 SUMMARY

This project work deals with image enhancement. Before image segmentation, the contrast of the image might be poor in which case, image segmentation will be adversely affected. Hence there is the need to enhance the image before segmenting it. One method of image enhancement is histogram equalization where the contrast levels are spread out throughout the grayscale levels of the image as discussed. The images used in this project work were Tolu1 and Tolu2. The input images were histogram-equalized to produce a better image.

5.2 CONCLUSION

From the result of this project work, it is clear that histogram equalization improved the contrast of images as a necessary step towards image segmentation. To improve this method, the thresholds for image segmentation can be made to be adaptive in which case it changes based on the two peak values of the histogram of the image.

5.3 RECOMMENDATION

It is therefore recommended that before image segmentation, the contrast has to be enhanced. There are various available techniques but it seems histogram equalization is easy to use and produces good result.

REFERENCES

Canny J. (1983). A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(6), 1986, 679-6987

Dwayne P. (2000). Image processing in C, second edition. R & D Publications 1601 West 23rd Street, Suite 200 Lawrence, Kansas 66046-0127.

Gonzales R. C. & Woods R. E. (2002). Digital Image Processing, 2nd ed. Reading, MA: Addison –Wesley, 2002.

Gurjeet K. S. & Rajneet K.(2013). Review on Recent Image Segmentation Techniques. International Journal of Computer Science and Engineering (IJCSE) Vol. 5 No. 02. Pp 109-112.

Image Noise of classifications in Remote Sensing, June 2011 [on line PDF]. http://www.dig.cs.gc.cuny.edu/seminars/IPCV/pres12.pdf

Kim S. & Kasper R. (2013). Applications of Convolution in Image Processing with MATLAB

Khotanzad, A. and Chen J. Y. (1989). "Unsupervised segmentation of textured images by edge detection in multidimensional feature," IEEE Trans. Pattern Analysis & Machine intelligence. vol. 11, no. 4, pp. 414–421.

Meenakshi G. & Seema (2013). Novel Approach for Removal of Gaussian and Salt-n-Pepper Noise Simultaneously. International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 7 pp 2714-2717.

McAndrew A., (2004). An Introduction to Digital Image Processing with Matlab. Notes for SCM2511 Image Processing 1, School of Computer Science and Mathematics Victoria University of Technology. pp 12.

Omwoyo J., Magana V. J., Mazana M., Maguya A., & Márquez M. M. (2007). Canny edge detector. Lappeenranta University of Technology.

Panjwani, D. K. and Healey, G., "Markov random field models for unsupervised segmentation of textured color images," IEEE Trans. Pattern Anal. Mach. Intell., vol. 17, no. 10, pp. 939–954.

Raju P.D.R, and Neelima G.(2012). Image segmentation by using histogram thresholding. IJCSET Vol 2, Issue 1, pp 776-779.

Rangarajan S, (2010). Algorithms for edge detection.

Salem S. A., Kalyankar N.V. & Khamitkar S.D (2010). Image Segmentation by Using Threshold Techniques. Journal of computing, volume 2, issue 5, pp 83-86.

Salem S. A., Kalyankar N.V. & Khamitkar S.D (2010). A comparative study and removal of noise from remote sensing image. International Journal of Computer Science Issues, Vol. 7, Issue. 1, No. 1. Pp 32-36.

Srinivasan K. S. & Ebenezer D.(2007). A new fast and efficient decision-based algorithm for removal of high-density impulse noises, *IEEE Signal Processing Letters*, vol. 14, no. 3, pp. 189–192.

Singh S., & Prakash N. R. (2012). Study and Analysis of Various Types of Noise affecting Image Quality. Fast Processing Peer Reviewed International Journals, India.

Young T.I,. Gerbrands J. J., & Vliet L.J (2007). Fundamentals of image processing. Version 2.3.

APPENDIX

```
% Histogram Equalization
clc; clear all;

%img = imread('img2.jpeg');
img = imread('Tolu1.jpg');

 if(size(img,3) > 1)
    img = rgb2gray(img);
end;


img = imresize(img,[256 256],'bicubic');
max_r = size(img,1);
max_c = size(img,2);
histogram = zeros([1 256]);
cumulative_hist = zeros([1 256]);
for r=1:max_r
    for c=1:max_c
        for count=1:256
            if(img(r,c) == count-1)
                histogram(count) = histogram(count) + 1;
                break;
            end
        end
```

```
    end
end

%find cumulative histogram
current_value = 0;
for count=1:256
    current_value = current_value + histogram(count);
    cumulative_hist(count) = current_value;
end
%find h = (cdf-cdf(min)/(MN-cdf(min)))*255
%this is the normalized cumulative histogram
normalized_hist = zeros([1 256]);
cdf_min = min(cumulative_hist);
for count=1:256
    normalized_hist(count) = cumulative_hist(count) - cdf_min;
    normalized_hist(count) = normalized_hist(count) / (
(max_r*max_c) - cdf_min);
    normalized_hist(count) = round(normalized_hist(count) *
255);
end
%replace the values with the given equalized values
equalized_image = zeros([max_r max_c]);
for r=1:max_r
    for c=1:max_c
```

```
        for count=1:256


            if(img(r,c) == (count-1))
                %we have got the value of intensity for this
pixel, replace
                %this with the equalized intensity
                equalized_img(r,c) = normalized_hist(count);
                break;
            end
        end
    end
end
subplot(2,2,1)
imshow(img);
title('Original Image');
subplot(2,2,2);
imhist(img);
title('Histogram of Original Image');
subplot(2,2,3);
imshow(uint8(equalized_img));
title('Histogram Equalized Image');
H=uint8(equalized_img);
subplot(2,2,4);
imhist(H);
title('Histogram of Histogram Equalized Image');
```

IMPLEMENTATION OF CONTRAST ENHANCEMENT IN IMAGE BASE ON HISTOGRAM EQUALIZATION

BY

BABAWALE OPEYEMI TOLULOPE
(MATRIC NO: CSC/11/0273)

BEIGN A PROJECT REPORT SUBMITTED

TO THE

DEPARTMENT OF COMPUTER SCIENCE, FACULTY OF SCIENCE,

FEDERAL UNIVERSITY OYE EKITI,

EKITI STATE, NIGERIA.

IN PARTIAL FUFILMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE (B.SC) DEGREE IN COMPUTER SCIENCE.

OCTOBER, 2015