CHAPTER-1

INTRODUCTION

1.1 Introduction

The problems of text detection and recognition in images and video have received increased attention in recent years, as indicated by the emergence of recent”robust reading” competitions in 2003, 2005, 2011, and 2013 along with bi-annual international workshops on camera-based document analysis and recognition (CBDAR)from 2005 to 2013. The emergence of applications on mobile devices, including the iPhone and Android platforms, which translate text into other languages in real time, has stimulated renewed interest in the problems. Several primary reasons for this trend exists including the demand of a growing number of applications. Text is one of the most expressive means of communications, and can be embedded into documents or into scenes as a mean of communicating information. This is done in the way that makes it noticeable and/or readable by others. The collection of massive amounts of street view” data is just one driving application. The second factor is the increasing availability of high performance mobile devices, with both imaging and computational capability. This creates an opportunity for image acquisition and processing anytime, anywhere, making it convenient to recognize text in various environments. The third is the advance in computer vision and pattern recognition technologies, making it more feasible to address challenging problems.

While many researchers view optical character recognition(OCR) as a solved problem, text detection and recognition in imagery possess many of the same hurdles as computer vision and pattern recognition problems driven by lower quality or degraded data. Ample room for research exists, as suggested by the low detection rates(often less than 80 percent) and recognition rates (often less than 60 percent) of state-of-the-art approaches. By contrast, OCR typically achieves recognition rates higher than 99 percent on scanned documents .Complex backgrounds, variations of text layout and fonts, and the existence of uneven illumination, low resolution and multilingual content present a much greater challenge than clean, well-formatted documents. Solving these problems requires the application of advanced computer vision and pattern recognition techniques.

1.2 Outline of the Thesis

For ease of understanding this book is divided in to 4 chapters. Chapter I is the Introduction and Chapter II describes in detail of all the techniques which are used for a project infrastructure & also gives the overview as well as the deep insight of the project in technical aspects. Chapter III describes the Requirement of the techniques along with detailed explanation of the steps for extracting the Image. Chapter IV has the Results and Chapter V concludes the project and provides the scope for further enhancement.

1.3 Scope of the Project

Although some standards have been established, to obtain character recognition and an overview from one camera but there will be a trade-off such as reduced low-light performance of a complete inability to recognize characters. Beware of all-embracing requirements such as, provide a color overview and identify characters in the text'. Nevertheless this project can be implemented on the upcoming advanced extraction techniques.

CHAPTER-2

DETAILS OF THE PROJECT UNDERTAKEN

2.1 Introduction

Text Extraction plays a major role in finding vital and valuable information. Text extraction involves detection, localization, tracking, binarization, extraction, enhancement and recognition of the text from the given image. These text characters are difficult to be detected and recognized due to their deviation of size, font, style, orientation, alignment, contrast, complex colored, textured background. Due to rapid growth of available multimedia documents and growing requirement for information, identification, indexing and retrieval, many researches have been done on text extraction in images. Several techniques have been developed for extracting the text from an image. The proposed methods were based on morphological operators, edge detection algorithm, Segmentation technique etc. All these techniques have their benefits and restrictions.

2.1.1 Block Diagram

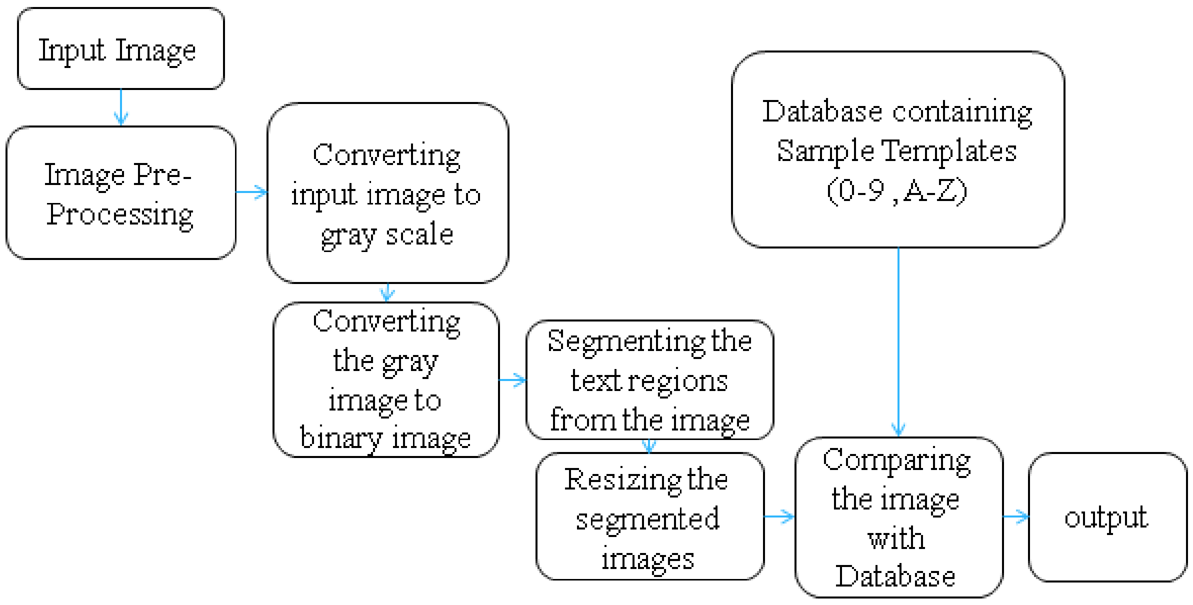

Input Image
Image Pre-Processing
Converting input image to gray scale
Converting the gray image to binary image
Segmenting the text regions from the image
Resizing the segmented images
Database containing Sample Templates (0-9 ,A-Z)
Comparing the image with Database
output

Fig2.1: block diagram

2.2 Weiner Filter

The most important technique for removal of blur in images due to linear motion or unfocussed optics is the Wiener filter. From a signal processing standpoint, blurring due to linear motion in a photograph is the result of poor sampling. Each pixel in a digital representation of the photograph should represent the intensity of a single stationary point in front of the camera. Unfortunately, if the shutter speed is too slow and the camera is in motion, a given pixel will be an amalgram of intensities from points along the line of the camera's motion. This is a two-dimensional analogy to

$$G(u,v)=F(u,v).H(u,v) \longrightarrow (1)$$

Where F is the Fourier transform of an "ideal" version of a given image, and H is the blurring function. In this case H is a sinc function: if three pixels in a line contain info from the same point on an image, the digital image will seem to have been convolved with a three-point boxcar in the time domain. Ideally one could reverse-engineer a Fest, or F estimate, if G and H are known. This technique is inverse filtering, as shown in the figure 2.2

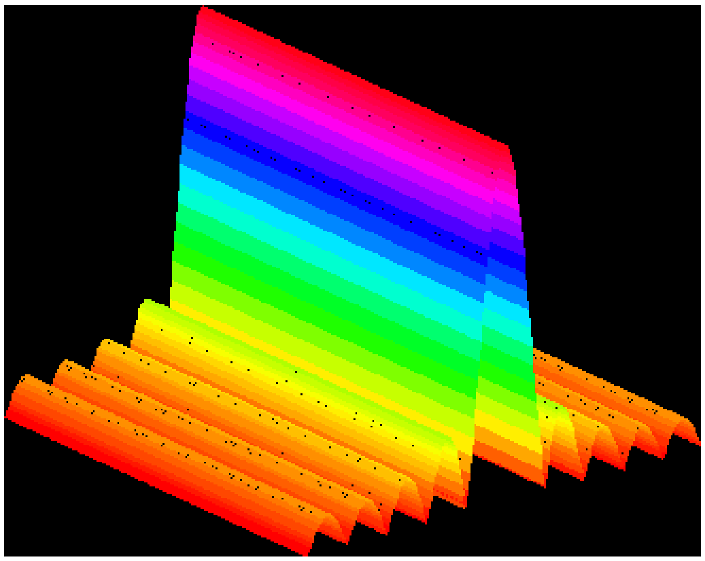


Fig 2.2: D Fourier Transform of Horizontal Blur

It should be noted that the image restoration tools described here work in a similar manner for cases with blur due to incorrect focus. In this case the only difference is in the selection of H**. [2]** The 2-d Fourier transform of H for motion is a series of sinc functions in parallel on a line perpendicular to the direction of motion; and the 2-d Fourier transform of H for focus blurring is the sombrero function, described elsewhere.

In the real world, however, there are two problems with this method. First, H is not known precisely. Engineers can guess at the blurring function for a given circumstance, but determination of a good blurring function requires lots of trial and error. Second, inverse filtering fails in some circumstances because the sinc function goes to 0 at some values of x and y. Real pictures contain noise which becomes amplified to the point of destroying all attempts at reconstruction of an Fest.

The best method to solve the second problem is to use Wiener filtering. This tool solves an estimate for F according to the following equation:

$$Fest(u,v) = |H(u,v)|^2.G(u,v)/(|H(u,v)|^2.H(u,v) + K(u,v)) \longrightarrow (2)$$

K is a constant chosen to optimize the estimate. This equation is derived from a least squares method. An example of Wiener filtering is given in below figure 2.3 .

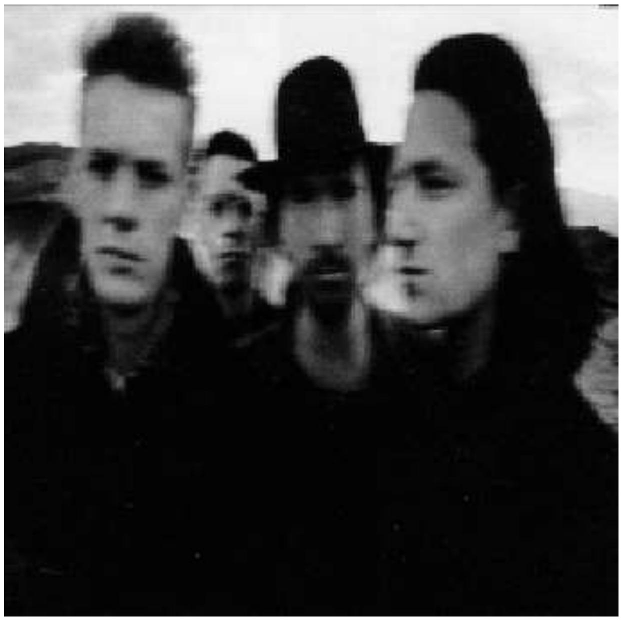
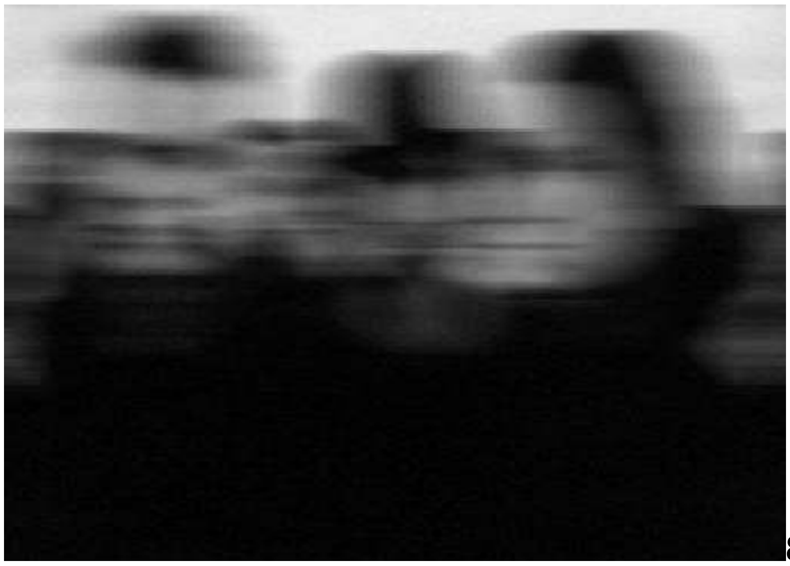


Fig2.3: Joshua Tree album by U2

An ideal version of the cover of the *Joshua Tree* album by U2. All examples used are 256x256 pixels, but the same principles apply if size is varied. Also, the examples are all gray scale but the principles are valid for color photos by applying filtering techniques separately to RGB elements as shown in figures 2.4 & 2.5.



8

Fig2.4: Joshua tree album with blurring

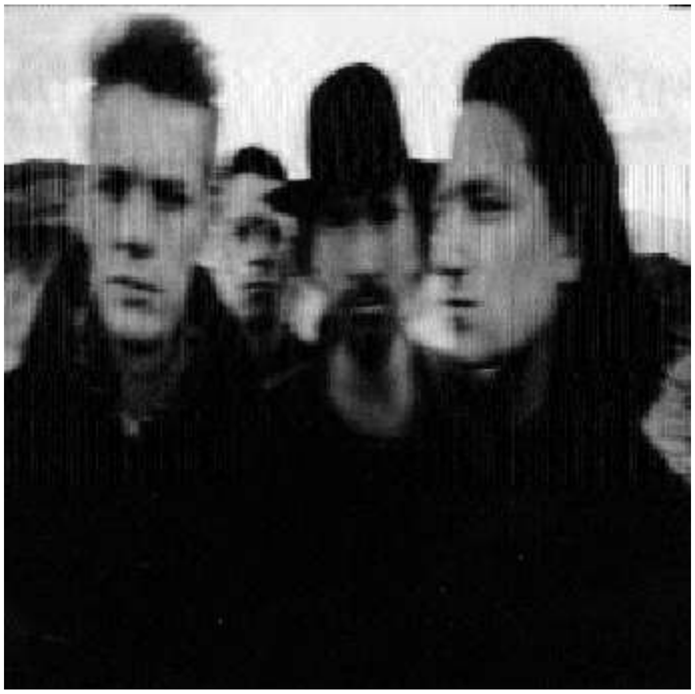


Fig2.5: Reconstructed photograph, e.g. f estimate, through Wiener filtering

In closing, it should be noted that Weiner filters are far and away the most common de-blurring technique used because it mathematically returns the best results. Inverse filters are interesting as a textbook starting point because of their simplicity, but in practice Wiener filters are much more common. [2] It should also be re-emphasized that Wiener filtering is in fact the underlying premise for restoration of other kinds of blur; and being a least-mean-squares technique, it has roots in a spectrum of other engineering applications.

2.3 RGB to Gray Conversion

It converts the true color image RGB to the gray scale intensity image. The rgb2gray function converts RGB images to gray scale by eliminating the hue (color or shade) and saturation information while retaining the luminance.

The use of color in image processing is motivated by two principal factors: First color is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, human can discern thousands of color shades and intensities, compared to about only two dozen shades of gray.**[3]** In RGB model, each color appears in its primary spectral components of red, green and blue. This model is based on Cartesian coordinate system.

Images represented in RGB color model consist of three component images. One for each primary, when fed into an RGB monitor, these three images combines on the phosphor screen to produce a composite color image. The number of bits used to represent each pixel in RGB space is called the pixel depth. Consider an RGB image in which each of the red, green and blue images are an 8- bit image. Under these conditions each RGB color pixel is said to have a depth of 24 bit.

2.3.1 True color image:

It is also known as an RGB image. A true color image is an image in which each pixel is specified by three values one each for the red, blue, and green components of the pixel scalar**[3].** M by-n-by-3 array of class uint8, uint16, single, or double whose pixel values specify intensity values. For single or double arrays, values range from [0, 1]. For uint8, values range from [0, 255]. For uint16, values range from [0, 65535].

2.3.2 Gray scale images:

Image formation using sensor and other image acquisition equipment denote the brightness or intensity I of the light of an image as two dimensional continuous function F(x, y) where (x,y) denotes the spatial coordinates when only the brightness of light is considered. Sometimes three-dimensional spatial coordinate are used. Image involving only intensity are called gray scale images.

Gray scale digital image is an image in which the value of each pixel is a single sample, that is, it carries only intensity information **[3].** Images of this sort, also known as black-and-white, are composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest.

Grayscale images are distinct from one-bit bi-tonal black-and-white images, which in the context of computer imaging are images with only the two colors, black, and white (also called bi level or binary images). Grayscale images have many shades of gray in between.

Grayscale images are often the result of measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum (e.g. infrared, visible light, ultraviolet, etc.), and in such cases they are monochromatic proper when only a given frequency is captured. But also they can be synthesized from a full color image; see the section about converting to grayscale

2.3.3 Resolution

Similar to one-dimensional time signal, sampling for images is done in the spatial domain, and quantization is done for the brightness values. In the Sampling process, the domain of images is divided into N rows and M columns **[3].** The region of interaction of a row and a

Coolum is known as pixel. The value assigned to each pixel is the average brightness of the regions. The position of each pixel was described by a pair of coordinates (xi, xj).

The resolution of a digital signal is the number of pixel is the number of pixel presented in the number of columns × number of rows. For example, an image with a resolution of 640×480 means that it display 640 pixels on each of the 480 rows. Some other common resolution used is 800×600 and 1024×728, among other. Resolution is one of most commonly used ways to describe the image quantity of digital camera or other optical equipment. The resolution of a display system or printing equipment is often expressed in number of dots per inch. For example, the resolution of a display system is 72 dots per inch (dpi) or dots per cm.

2.3.4 Gray levels:

Gray levels represent the interval number of quantization in gray scale image processing. At present, the most commonly used storage method is 8-bit storage. There are 256 gray levels in an 8 bit gray scale image, and the intensity of each pixel can have from 0 to 255, with 0 being black and 255 being white we. Another commonly used storage method is 1-bit storage.

There are two gray levels, with 0 being black and 1 being white a binary image, which, is frequently used in medical images, is being referred to as binary image . As binary images are easy to operate, other storage format images are often converted into binary images when they are used for enhancement or edge detection. Below figure 2 shows a typical gray scale image with red, blue, green images, respectively as shown in fig 2.6.

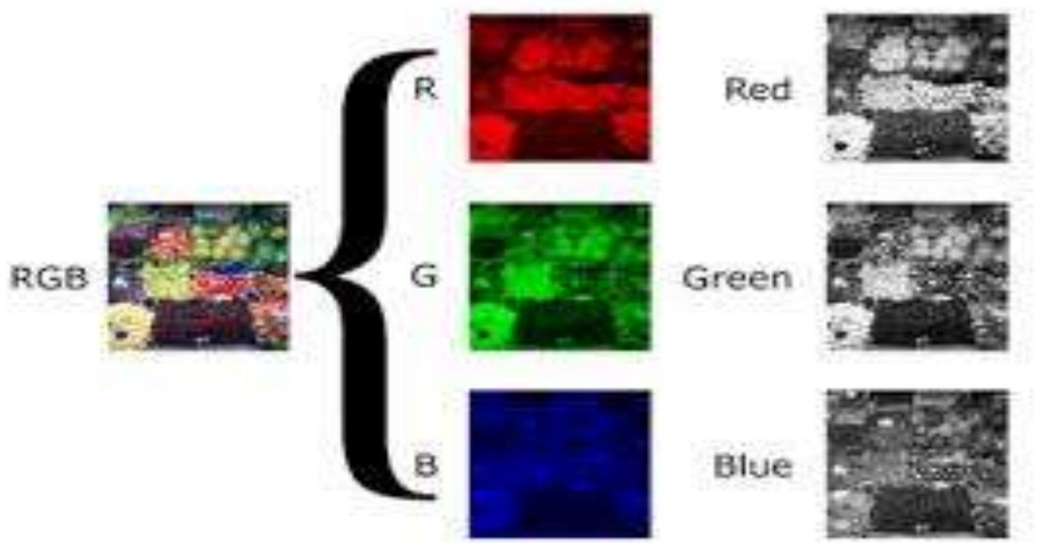


Fig2.6 : Conversion of original image RGB to Gray color

2.3.5 RGB color model:

In RGB color model, each color appears in its primary spectral components of red, green and blue. The color of a pixel is made up of three components; red, green, and blue (RGB), described by their corresponding intensities **[3].** Color components are also known as color channels or color planes (components). In the RGB color model, a color image can be represented by the intensity function.

$$\mathbf{I_{RGB}= (FR, FG, FB)} \longrightarrow (3)$$

Where,

FR(x,y) is the intensity of pixel (x,y) in the red channel,
FG(x,y) is the intensity of pixel (x,y) in the green channel,
FB(x,y) is the intensity of pixel (x,y) in the blue channel.

The intensity of each color channel is usually stored using eight bits, which indicates that the quantization level is 256. That is, a pixel in a color image requires a total storage of 24 bits. A 24 bit memory can express as 224 =256×256×256=16777216 distinct colors.

The number of colors should adequately meet the display effect of most images. Such images may be called true color images, where information of each pixel is kept by using a 24-bit memory.

Above figure shows the images of a 24-bit color RGB, three channels (component) and corresponding pixel information image. If only the brightness information is needed, color images can be transformed to gray scale images.
The transformation can be made by using proposed equation.

$$\mathbf{Iy=0.333FR+0.5FG+0.1666FB} \longrightarrow (4)$$

Where,

FR, FG and FB are the intensity of R, G and B component respectively and
Iy is the intensity of equivalent gray level image of RGB image.

In this way we can calculate all the gray level by using above transformation and also calculate the percentage error.

2.4 Thresholding:

- In thresholding, the color-image or gray-scale image is reduced to a binary image.
- The simplest property that pixels in a region can share is intensity. So, a natural way to segment such regions is through thresholding, the separation of light and dark regions.
- Simplest method of image segmentation is called the thresholding method. This method is based on a clip-level (or a threshold value) to turn a gray-scale image into a binary image. There is also a balanced histogram thresholding.
- 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. (What you want to do with pixels at the threshold doesn't matter, as long as you're consistent.).

If g(x,y) is a thresholded version of f(x,y) at some global threshold T,

$$g(x,y) = \begin{matrix} 1 & \text{if} \quad f(x,y) \geq T \\ 0 & \text{otherwise} \end{matrix} \longrightarrow \quad (5)$$

A process of creating a black-and-white image out of a grayscale image consisting of setting exactly those pixels to white whose value is above a given threshold, setting the other pixels to black as shown in fig 2.7&2.8.

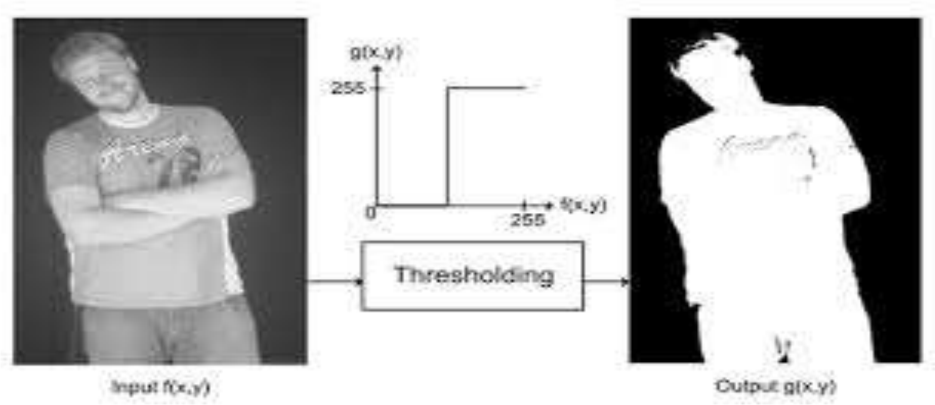


Fig2.7: Output image after Thresholding.

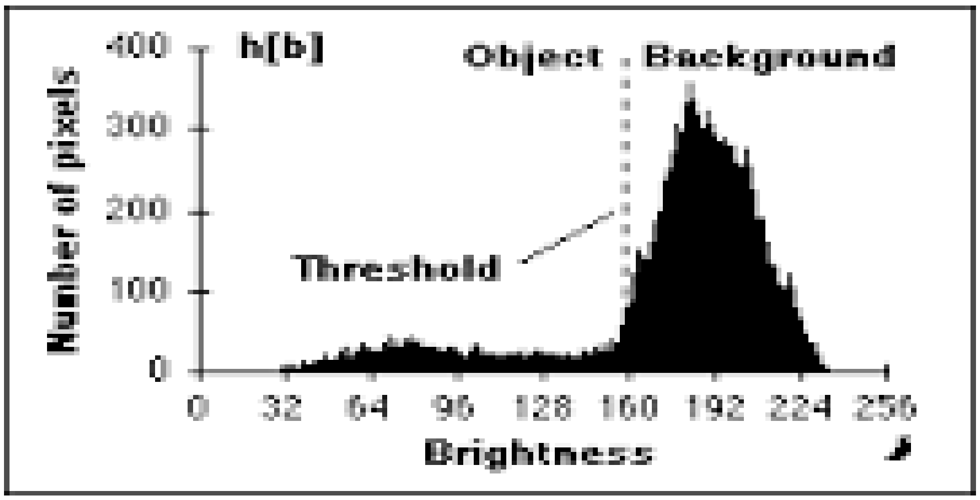


Fig2.8 : The variation of intensity levels after Thresholding with input image.(From thresholding and binarization document)

The purpose of thresholding is to extract those pixels from some image which represent an object (either text or other line image data such as graphs, maps**). [2]** Though the information is binary the pixels represent a range of intensities.

Thus the objective of binarization is to mark pixels that belong to true foreground regions with a single intensity and background regions with different intensities.

2.4.1 Types of Thresholding:

There are two types of thresholding algorithm

1. Global thresholding algorithms: In global thresholding, a single threshold for all the image pixels is used. When the pixel values of the components and that of background are fairly consistent in their respective values over the entire image, global thresholding could be used

2. Local or adaptive thresholding algorithms: In adaptive thresholding, different threshold values for different local areas are used.

2.4.2 Problems with Thresholding:

The major problem with Thresholding is that we consider only the intensity, not any relationships between the pixels **[2].** When we use Thresholding, we sometimes lose too much of the region and sometimes get too many extraneous background pixels.

2.4.3 Binarization

Binarization is the process of converting a pixel image to a binary image. In the old days binarization was important for sending faxes. These days it's still important for things like digitalizing text or segmentation as shown in fig 2.9.

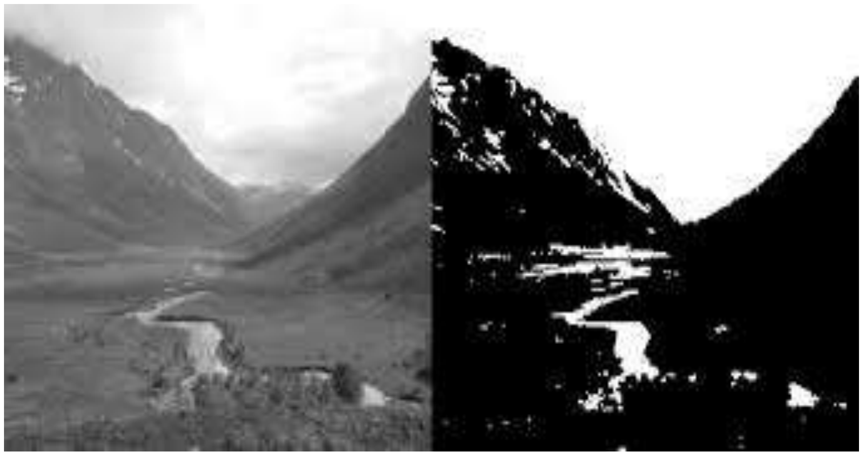


Fig2.9: Image after thresholding

2.5 Connected components

In binary valued digital imaging, a pixel can either have a value of 1 -when it's part of the pattern- , or 0 -when its part of the background- i.e. there is no gray scale level. (We will assume that pixels with value 1 are black while zero valued pixels are white).

In order to identify objects in a digital pattern, we need to locate groups of black pixels that are "connected" to each other. In other words, the objects in a given digital pattern are the connected components of that pattern.**[4]** In general, a connected component is a set of black pixels, P, such that for every pair of pixels p_i and p_j in P, there exists a sequence of pixels p_i, ..., p_j such that:

a) All pixels in the sequence are in the set P i.e. are black, and every 2 pixels that are adjacent in the sequence are "neighbors"

As a result, an important question arises: When can we say that 2 pixels are "neighbors"? Since we are using square pixels, the answer to the previous question is not trivial. The reason for that is: in a square tessellation, pixels either share an edge, a vertex, or neither. There are 8 pixels sharing an edge or a vertex with any given pixel; these pixels make up the Moore neighborhood of that pixel. Should we consider pixels having only a common vertex as "neighbors"? Or should 2 pixels have a common edge in order for them to be considered "neighbors"?

This gives rise to 2 types of connectedness, namely: 4-connectivity and 8-connectivity.

4-Connectivity

When can we say that a given set of black pixels is 4-connected? First, we have to define the concept of a 4-neighbor (also known as a direct-neighbor):

Definition of a 4-neighbor:

A pixel, Q is a 4-neighbour of a given pixel, p, if Q and P share an edge the 4-neighbours of the pixel P are: P2, P4, P5, and P6 as shown in the figure 2.10.

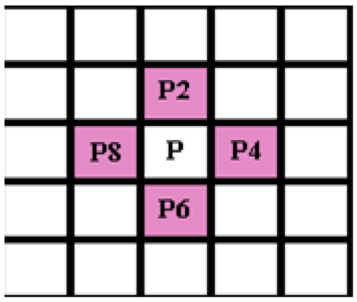


Fig: 2.10 4 connectivity

Definition of a 4-connected component:

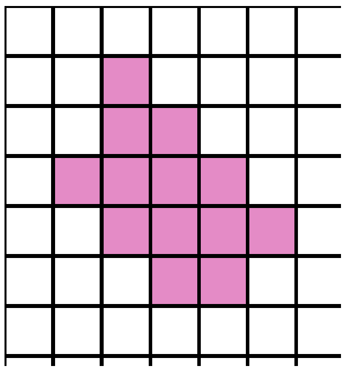
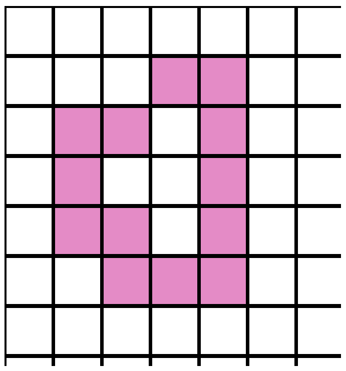
A set of black pixels, P, is a 4-connected component if for every pair of pixels p_i and p_j in P, there exists a sequence of pixels p_i, ..., p_j such that:

a) All pixels in the sequence are in the set P i.e. are black, and

b) Every 2 pixels that are adjacent in the sequence are 4-neighbors

Examples of 4-connected patterns:

The following diagrams are examples of patterns that are 4-connected:

8-Connectivity

When can we say that a given set of black pixels is 8-connected? First, we have to define the concept of an 8-neighbor (also known as an indirect-neighbor):

Definition of an 8-neighbor:

A pixel, Q, is an 8-neighbor (or simply a neighbor) of a given pixel, P, if Q and P either share an edge or a vertex.

The 8-neighbors of a given pixel P make up the Moore neighborhood of that pixel.

Definition of an 8-connected component:

A set of black pixels, P, is an 8-connected component (or simply a connected component) if for every pair of pixels p_i and p_j in P, there exists a sequence of pixels $p_i, ..., p_j$ such that:

a) all pixels in the sequence are in the set P i.e. are black, and

b) every 2 pixels that are adjacent in the sequence are 8-neighbors

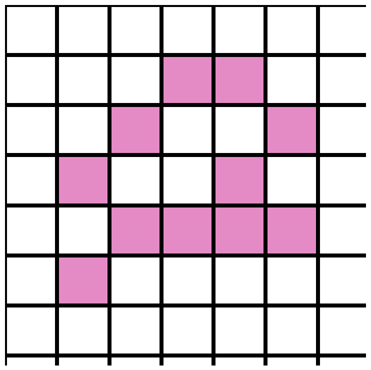
NOTE

All 4-connected patterns are 8-connected i.e. 4-connected patterns are a subset of the set of 8-connected patterns.

On the other hand, an 8-connected pattern may not be 4-connected.

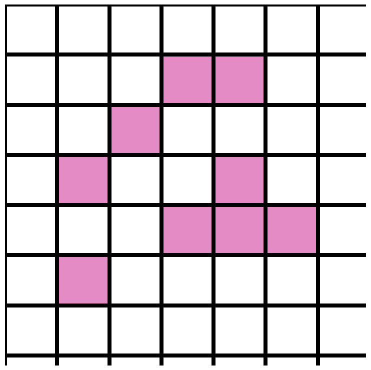
Example of 8-connected pattern:

The diagram below is an example of a pattern that is 8-connected but not 4-connected:



Example of a pattern that's not 8-connected:

The diagram below is an example of a pattern that is not 8-connected i.e. is made up of more than one connected component (there are 3 connected components in the diagram below):



2.6 Image Segmentation

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super pixels)**.[4]** 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.

Image segmentation is typically used to locate objects and boundaries in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (see edge detection). Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture.

Adjacent regions are significantly different with respect to the same characteristic(s). When applied to a stack of images, typical in medical imaging, the resulting contours after image segmentation can be used to create 3D reconstructions with the help of interpolation algorithms like marching cubes.

Segmentation is having its significance in various object identification and selection applications. The accuracy of these applications depends on accurate feature selection or segmentation. In this an exploration to the segmentation approach is presented in a generalized way. Here the segmentation approaches are classified in edge based segmentation, region based segmentation and threshold based segmentation.

The term image segmentation refers to the partitioning of an image into a set of regions that cover it. The goal in many tasks is for the regions to represent meaningful areas of the image, such as the crops, urban areas, building areas and forests of a satellite image.

Image segmentation varies from filtering of noisy images, medical applications (locating tumors and other pathologies, planning of treatments, diagnosis, etc.), locating objects in satellite images (Beach, building, river, forest, etc.), face recognition etc. The way of segmenting the image is shown with an example in figure 2.10.

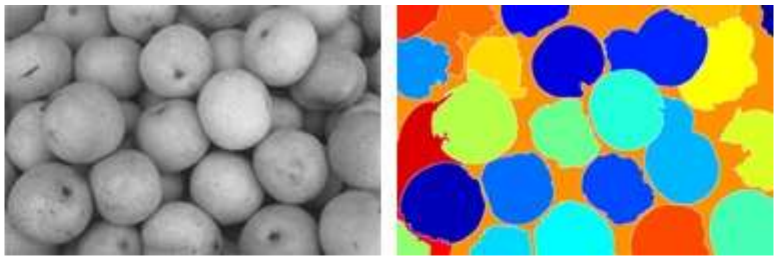


Fig2.11 : Image segmentation

2.6.1 Segmentation Techniques

Image segmentation is an important stage in image processing. There are several segmentation approach existing

All segmentation approaches are based on two principles given below:

1. **Discontinuity principle:** In this, subdivision of images is carried out on the basis of abrupt changes in intensity or gray levels of an image.

2. **Similarities principle:** In this, subdivision of images is carried out on the basis of similarities in intensity or gray level of an image.

2.6.2 Image Segmentation Methods

Image segmentation methods are as follows:

Edge based segmentation

Edge detection is well developed field in image processing. Edge represents boundaries of an object. Edge detection is used in image segmentation for detection of object boundaries.

Edges are detected when there is an abrupt change in intensity or change in brightness of image. Methods for edge based segmentation are:

1. Gray histogram Method

The quality of edge detection will depends on fitness of T (threshold value). In this method segmentation depends on selection of maximum and minimum threshold value and this selection of threshold is very difficult because the gray histogram is uneven for the images which are affected by noise.

2. Gradient Based method

Gradient method is used to detect the edge with the help of local maxima and local minima. To find local maxima and local minima we use first derivative and gradient is first derivative of image**.[4]** When there is sharp change in intensity and there is some image noise, Gradient based method works well, and the result of segmentation is adaptive to the direction of gradient. There are mainly three edge detection operator Laplace operator, canny operator, and laplacian of Gaussian (LOG) operator canny operator gives best result.

Region based segmentation

This method is based on continuity. Regions of an image are group of number of connected pixels with same gray level. In region based segmentation, each pixel is assigned to a particular region. In this method pixels belonging to an object are grouped together. This method divides the whole image into sub regions on the basis of common patterns (i.e intensity value). Region based segmentation is simple then edge based segmentation.

This type of segmentation is kind of iterative algorithm. Region based segmentation basically includes following methods:

1. Region growing method:

As it is suggested by its name, region growing is a procedure that groups pixels into sub region or sub regions into large regions based on some predefined criteria. The procedure for this method is as follows:

1. Select a set of seed particles in original image.
2. Select a set of criteria for determining similar seeds based on properties such as grey level intensity or color and then set up a stopping rule.
3. After selection of seeds, grow regions by appending each seed to its neighboring pixels that have similar properties of the seed (such as intensity, gray level or color).
4. Stop the region growth when no more pixels satisfy the similarity criteria.

2. Region Splitting and Merging:

Region splitting and merging method is opposite of region growing method. It is top down approach.**[4]** The way of approaching is shown in figure 23. It basically divides an image into a set of arbitrary, unconnected regions and then merges and/or split the regions in an attempt to satisfy the conditions of reasonable image segmentation. Region splitting and merging is usually implemented with theory based on quad tree data as shown in figure 2.11.

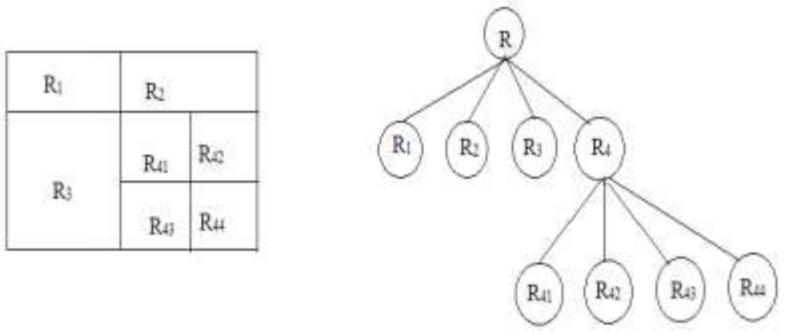


Fig2.12 : Quad tree

Thresholding based segmentation

This method of segmentation is very powerful method. This method is used to segment foreground or an object from background. This method depends on the properties of an image. For proper detection either foreground is lighter then background or vice-versa. To find an object from background is basically done by selecting a threshold value (i.e. T).Let us take a pixel(x, y),

a) If pixel(x, y)'s value is greater than or equal to threshold value i.e. f(x, y)>= T then object is considered as foreground.

b) If pixel(x, y)'s value is greater than or equal to threshold value i.e. $f(x, y) <= T$ then object is considered as background.

1. Global thresholding:

This method is used when the intensity distribution between objects of foreground and background is different. That is, this method uses a fixed threshold value T. Based on T foreground and background pixels are identified when the foreground and background objects are distinct.

2. Local thresholding:

Local thresholding is used when global thresholding does not work. That is, in this method image is divided into sub images and each sub image is thresholded separately. In this method various thresholds are selected one for each sub image.

2.7 Character Recognition

It is employed for the purpose of conversion of images of text into characters. Character recognition is now used to compare the each individual character against the complete alphanumeric database using template matching.

The matching process moves the template image to all possible positions in a larger source image and computes a numerical index that indicates how well the template matches the image in that position. Matching is done on a pixel by pixel basis.

The template is of size 42×24 as shown in Fig.26.Since the template size is fixed, it leads to accurate recognition. Extraction and recognition of Text system basically consist of 3 main processing steps such as:

- Detection of Text
- Segmentation of text characters
- Recognition of each character.

Among this, character segmentation is a most challenging task, as the accuracy of the character recognition relies on the accuracy of the character segmentation. Problems of different lighting condition, adhesion, fracture, rivet, rotation degrades the accuracy of the character - segmentation.

So in order to overcome these problems and uplift the accuracy of character segmentation various algorithms are developed for this work as shown in the figure 2.13.

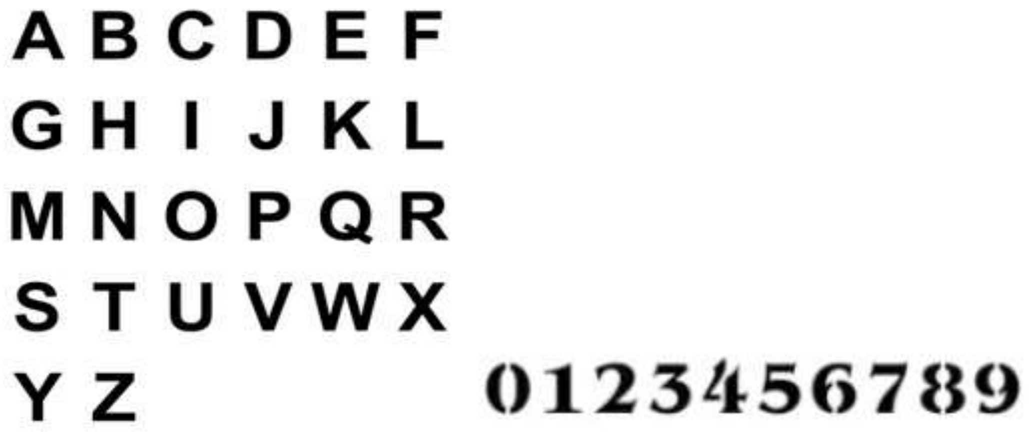


Fig2.13: Templates Used for Template Matching

2.8 Correlation

Correlation and are basic operations that we will perform to extract information from images. They are in some sense the simplest operations that we can perform on an image, but they are extremely useful. Moreover, because they are simple, they can be analyzed and understood very well, and they are also easy to implement and can be computed very efficiently.**[5]** Our main goal is to understand exactly what correlation and do, and why they are useful. We will also touch on some of their interesting theoretical properties

2.8.1 Matching with Correlation

Part of the power of correlation is that we can use it, and related methods, to find locations in an image that are similar to a template.**[5]** To do this, think of the filter as a template; we are sliding it around the image looking for a location where the template overlaps the image so that values in the template are aligned with similar values in the image. First, we need to decide how to measure the similarity between the template and the region of the image with which it is aligned. A simple and natural way to do this is to measure the sum of the square of the differences between values in the template and in the image. This increases as the difference between the two increases.**[5]** For the difference between the filter and the portion of the image centered at x, we can write this as: ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) ( ) = − − = −= =− =− = + + − + − + = + + − + N i N N i N N i N N i N N i N F i I x i F Ii x i F i I x i F i I x i F Ii x i 2)(( ) 2 2 2 2 2 2 As shown, we can break the Euclidean distance into three parts. The first part depends only on the filter. This will be the same for every pixel in the image. The second part is the sum of squares of pixel values that overlap the filter.

And the third part is twice the negative value of the correlation between F and I. We can see that, all things being equal, as the correlation between the filter and the image increases, the Euclidean distance between them decreases.**[5]** This provides an intuition for using correlation to 1 4 9 16 25 36 . . . 1 1/2 4 6 8 10 12 . . . match a template with an image. Places where the correlation between the two is high tend to be locations where the filter and image match well. This also shows one of the weaknesses of using correlation to compare a template and image.

CHAPTER-3

DESCRIPTION OF THE PROJECT

3.1 Introduction

This project presents a method of Extraction and Character recognition. The whole system into three following steps:

1- Finding location of text in the image.
2- Segmentation or cutting image to character's images.
3- Character recognition or convert character's images to final distinguished characters among them.

Template matching or matrix matching, is one of the most common classification methods. Here individual image pixels are used as features. Classification is performed by comparing an input character with a set of templates (or prototypes) from each character class. Each comparison results in a similarity measure between the input characters with a set of templates. One measure increases the amount of similarity when a pixel in the observed character is identical to the same pixel in the template image. If the pixels differ the measure of similarity may be decreased. After all templates have been compared with the observed character image, the character's identity is assigned the identity of the most similar template. Template matching is a trainable process as template characters can be changed.

3.2 Working of the Project

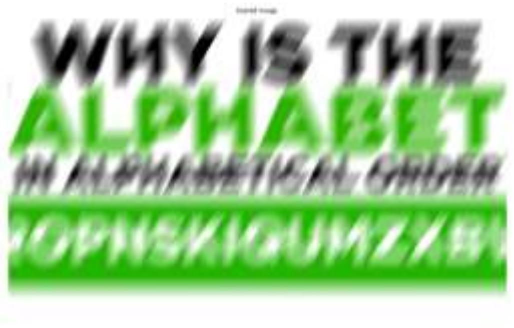
3.2.1 Blurred Image Captured By Camera

The Blurred image whose text is to be identified is captured using digital camera, which is given as input image to our extraction and recognition system. The blurred image is shown in figure 27.

3.2.2 Pre-Processing:

Pre-processing mostly is necessary to facilitate further high performance recognition, in this proposed methodology, the character is binarized and the noise is eliminated in the pre-processing stage. The pre-processed image is shown in figure 27.1.

Pre-processing is carried out on the captured image to improve the quality of the image so that the main processing on the image becomes easier. After the pre-processing, well contrast enhanced and changing the colour image in to gray now it is to feed into the main body of Text Extraction system as shown in the figures 3.1&3.2.



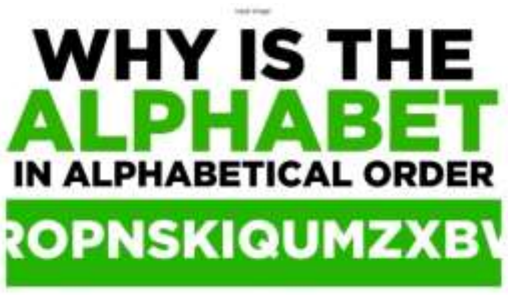


Fig3.1: Blurred image

Fig 3.2: Pre-processed image

3.2.3 RGB to Gray Conversion

We have taken colour image of car clearly showing its License plate for experiment. We firstly convert this RGB colour input image to a 256 gray scale image using formula (1).

The gray image is shown in figure 3.3.

$$\textbf{GREY=J(:,:,1)+J(:,:,2)+J(:,:,3)} \longrightarrow (6)$$

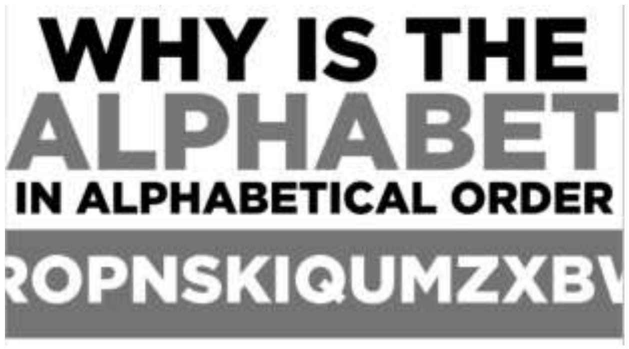


Fig3.3: Gray Scale Image

The grey-scale could take every pixel of the picture to a number between 0-255 and the purpose of the binarization is to take every pixel into the number 0 or 255. To remove the tonal variation between Red, Green and Blue channels of input images and converting it into grey scale flatness to a single hue.

Filtering

Filtering can be done in two ways one is spatial filtering and second one is frequency domain. Farther the spatial filtering can be done in two ways first is mean filtering and second is median filtering.

Median Filtering:

It is inevitable for containing noises of original image. We use median filtering to eliminate the noises. Using median filtering not only can eliminate the noises, but also make the high frequency more concentrated. There by, it is beneficial for us to detect the edges in images. The salt and pepper characteristic of Text image is presented, So Median filter is best known choice for removing such noise.

3.2.4 Binarization

To extract the text from a colour image it is compulsion that the image must be converted to binary form. Binarization of the image is done with the help of Thresholding method. The binary image is shown in figure 3.4.

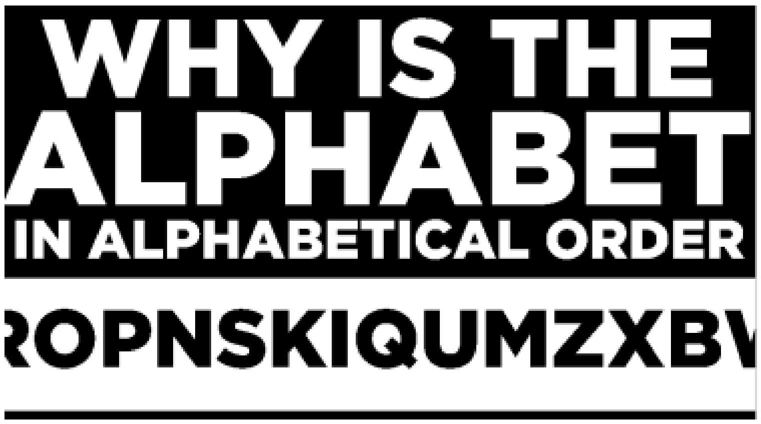


Fig3.4: Binary image

3.2.5 Character Segmentation

Character segmentation is an important stage in text extraction and recognition system. There are many factors that cause the character segmentation task difficult, such as image noise, plate frame, rivet, and rotation and illumination variance. Object segmentation is an essential task in computer vision and object recognitions.

Image segmentation is the process of partitioning a digital image into multiple regions or sets of pixels. These partitions represent different objects in the image, usually having the same texture or colour.

Segmentation is quite essential to image feature extraction and subsequent classification of the resultant features. This step is very significant due to overlapping characters that form the Text image.

There are three main forms of characters that are overlapping vertically, ligature, diacritics, horizontal overlap, and two connected characters. The task will be more difficult for those different forms of which are joined.

CHAPTER-4

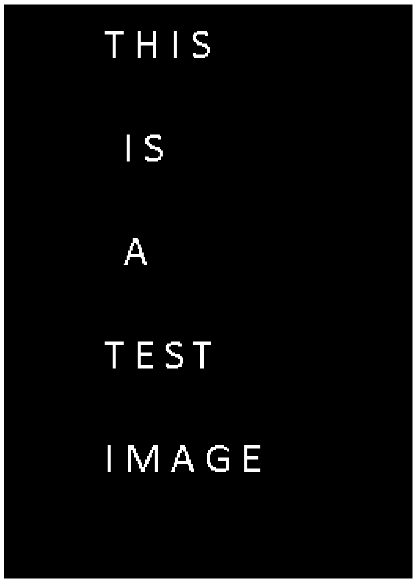
RESULTS

4.1 RESULT

1. **Captured Blur Image**

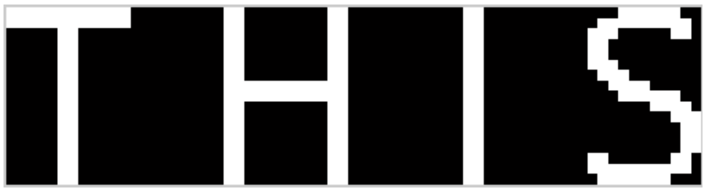
2. **RGB to Gray conversion**

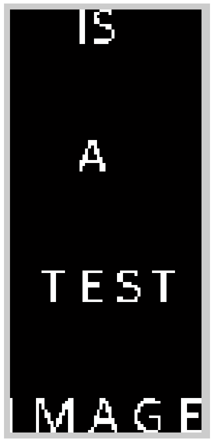
3. Binarization



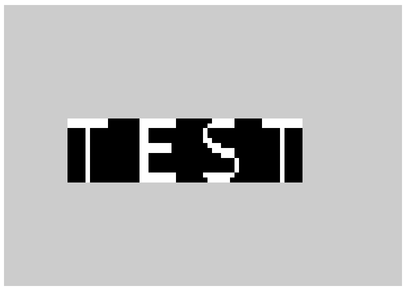
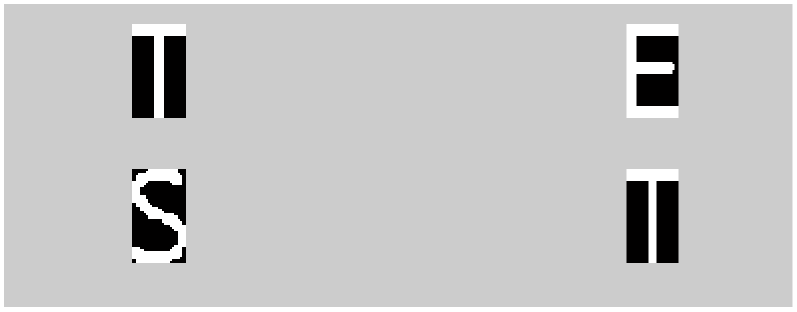
The gray Scale image is then converted to binary image using Otsu method thresholding and the text extraction of the text in binary image is done using connected components method.

4. Text Extraction



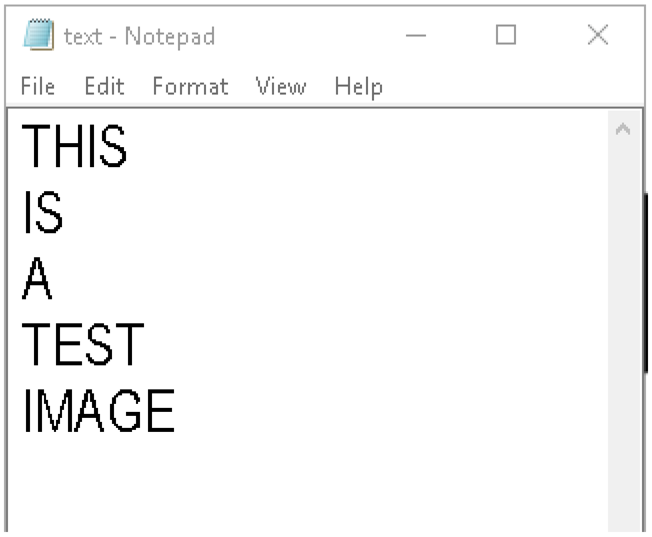


Here, we used 8- connectivity and extracted the text.

Similarly, Using 8-connectivity in the extracted 1st line we exrtracted the text from the line.

5. Recognizing text and Printing



THIS
IS
A
TEST
IMAGE

The Extracted characters are then compared with the existing database using correlation technique and then printed to a text editor.

4.2 Applications

- Convert billboards in to text format for translation.
- Data entry for business documents e.g. cheque, passports, invoice etc.
- Make electronic text of printed documents e.g. Google books.
- Converting handwriting into real time text (pen computi

4.3 Conclusion:

In this project we have presented a comprehensive approach for the detection of text region and recognition of texts from images. A great deal of work will have to be done to make the system more efficient The proposed method is tested with various clusters of images, images with the caption text. All related methods which are given in references are analyzed and the drawbacks are tried to reduce and thereby getting an improved version of the previous works. In this work, we obtain reduced noise levels and comparatively high precision rate of recognized text from an image.

.4.4 Future Scope:

This project can further be extended to recognize many font styles and blurring techniques can be enhanced. We can also extend the usage of this algorithm for machine learning and analysis, with further enhancement we can deploy this project in auto or self-driving cars to read the road signs and act accordingly.

4.5 REFERENCES:

[1] Jain, A.K. and Yu, B., Automatic text location in images *Pattern Recognition*vol. 31, no. 12, pp.2055-2076, 1998.

[2]. Rafael and Richard,2009, “Digital Image Processing”, 3rd ed. Prentice-Hall Inc.

[3] Ohya, J., Shio, A., and Akamatsu, A., Recognition of] characters in scene images. *IEEE Transactions on PatternAnalysis and Machine Intelligence*, vol. 16, no. 2, pp. 214-220, 1994.

[4] Sato, T., Kanade, T., Hughes, E.K., and Smith, M.A., Image OCR for digital news archives. *IEEE Int. Workshop onContent-Based Access of Image Database,* 1998.

[5] Class Notes for CMSC 426, Fall 2005 David Jacobs

[6] S. J. Chapman. MATLAB Programming for Engineers. Thomson, 2004.
.
[7] The MathWorks Inc. MATLAB 7.0 (R14SP2). The MathWorks Inc., 2005

APPENDIX

IMAGE PROCESSING

1.1 Introduction

Information carrying function of time is called signal. Real time signals can be audio (voice) or video (image) signals. Still video is called an image. Moving image is called a video. Difference between digital image processing and signals and systems is that time graph is not there in DIP. X and Y coordinates in DIP are spatial coordinates. Time graph is not there because photo doesn't change with time.

Image:

An image is defined as a two dimensional function f(x, y) where x and y are spatial coordinates and the amplitude 'f' at any point (x, y) is known as the intensity

of image at that point.

Pixel:

A pixel (short for picture element) is a single point in a graphic image. Each such information element is not really a dot, nor a square but an abstract sample. Each element of the above matrix is known as a pixel where dark = 0 and light = 1.

A pixel with only 1 bit will represent a black and white image. If the number of bits is increased then the number of gray levels will increase and a better picture quality is achieved.

All naturally occurring images are analog in nature. If the number of pixels are more then the clarity of the image is better. An image is represented as a matrix in DIP. In DSP we use only row matrices. Naturally occurring images should be sampled and quantized to get a digital image. A good image should have 1024*1024 pixels which is known as

1k * 1k = 1M pixel

1.2 Fundamental steps in DIP :

1. Image acquisition: Digital image acquisition is the creation of digital images typically from a physical object.**[6]** A digital image may be created directly from a physical scene by a camera or similar device. Alternatively it can be obtained from another image in an analog medium such as photographs, photographic film, or printed paper by a scanner or similar device.

Many technical images acquired with to monographic equipment, side-looking radar, or radio telescopes are actually obtained by complex processing of non-image data.

2. Image enhancement: The process of image acquisition frequently leads to image degradation due to mechanical problems, out-of-focus blur, motion, inappropriate illumination and noise. The goal of image enhancement is to start from a recorded image and to produce the most visually pleasing image.

3. Image restoration: The goal of image restoration is to start from a recorded image and to produce the most visually pleasing image. The goal of enhancement is beauty.**[6]** The goal of restoration is truth. The measure of success in restoration is usually an error measure between the original and the estimate image. No mathematical error function is known that corresponds to human perceptual assessment of error.

4. Color image processing: Color image processing is based on that any color can be obtained by mixing 3 basic colors red, green and blue. Hence 3 matrices are necessary each one representing each color.

5. Wavelet and multi resolution processing : Many times a particular spectral component occurring at any instant can be of particular interest. In these cases it may be very beneficial to know the time intervals these particular spectral components occur.**[7]** For example, in EEGs the latency of an event-related potential is of particular interest. Wavelet transform is capable of providing the time and frequency information simultaneously, hence giving a time-frequency representation of the signal.

Although the time and frequency resolution problems are results of a physical phenomenon (the Heisenberg uncertainty principle) and exist regardless of the transform used, it is possible to any signal by using an alternative approach called the multi resolution analysis (MRA). MRA analyzes the signal at different frequencies with different resolutions. MRA is designed to give good time resolution and poor frequency resolution at high frequencies and good frequency resolution and poor time resolution at low frequencies.

6. Compression: Image compression is the application of data compression on digital images. Its objective is to reduce redundancy of the image data in order to be able to store or transmit data in an efficient form.

7. Morphological processing: Morphological processing is a collection of techniques for DIP based on mathematical morphology. Since these techniques rely only on the relative ordering of pixel values not on their numerical values they are especially suited to the processing of binary images and grayscale images.

8. Segmentation: In the analysis of the objects in images it is essential that we can distinguish between the objects of interest and "the rest". This latter group is also referred to as the background**.[6]** The techniques that are used to find the objects of interest are usually referred to as segmentation techniques.

1.3 Properties of light:

1. Radiance: The total energy that flows from a light source is called radiance. Units are watts. Example of sources are sun, bulb etc.

2. Luminance: The amount of energy that an observer perceives from the source is called luminance. Units are lumens. Example is seeing the sun with black glasses.

3. Brightness: Brightness is an attribute of visual perception in which a source appears to emit a given amount of light. It has no units as it is practically impossible to measure.

1.4 Representing digital images:

The result of sampling and quantization of an image is an array of numbers according to x and y coordinates which is nothing but a matrix of m rows and n columns. Columns and rows depend on number of samples**.[6]** Brightness depends on number of bits used to represent each pixel. For an image of size m rows and n columns total number of pixels = m*n

If each pixel is represented using k bits then total amount of memory (in bits) required to store the image = m*n*k

A 1M pixel size image requires 1024 rows and 1024 columns. If each pixel is represented using 8 bits then total amount of memory required to store the image is

1k*1k*1 bytes

1.5 Resolution:

Resolution is classified into 2 types

1. Spatial
2. Gray level

1. Spatial Resolution:

It is the smallest discernable detail in the image. Consider vertical lines of width 'w' with spaces between also having width 'w'. A line pair consists of one such line & its adjacent space. Hence width of line pair = 2w and there are 1/(2*w) line pairs per unit distance. Thus spatial resolution is the number of discernable line pairs per unit distance.

Example: 100 line pairs/unit distance

2. Gray level resolution:

It is the smallest discernable change in gray level. The number of gray levels should be a power of 2. The commonly used number of gray levels is 256.

$L = 256$

$2^k = 256$

$k = 8$ (8 bits per pixel)

1.6 MATLAB details:

1. Basic flow control and programming language
2. How to write scripts (main functions) with matlab
3. How to write functions with matlab
4. How to use the debugger
5. How to use the graphical interface
6. Examples of useful scripts and functions for image processing

After learning about matlab we will be able to use matlab as a tool to help us with our maths, electronics, signal & image processing, statistics, neural networks, control and automation.

Matlab resources:

Language:

High level matrix/vector language with

- Scripts and main programs
- Functions
- Flow statements (for, while)
- Control statements (if,else)
- data structures (struct, cells)
- input/ouputs (read,write,save)
- Object oriented programming.
-

Environment:

- Command window.
- Editor
- Debugger
- Profiler (evaluate performances)

Mathematical libraries:

- Vast collection of functions

API:

- Call c function from matlab
- Call matlab functions from c

Scripts and main programs:

In matlab, scripts are the equivalent of main programs. The variables declared in a script are visible in the workspace and they can be saved. Scripts can therefore take a lot of memory if you are not careful, especially when dealing with images. To create a script, you will need to start the editor, write your code and run it.

1.7 MATLAB Functions:

imread: Read images from graphics files.

Syntax:

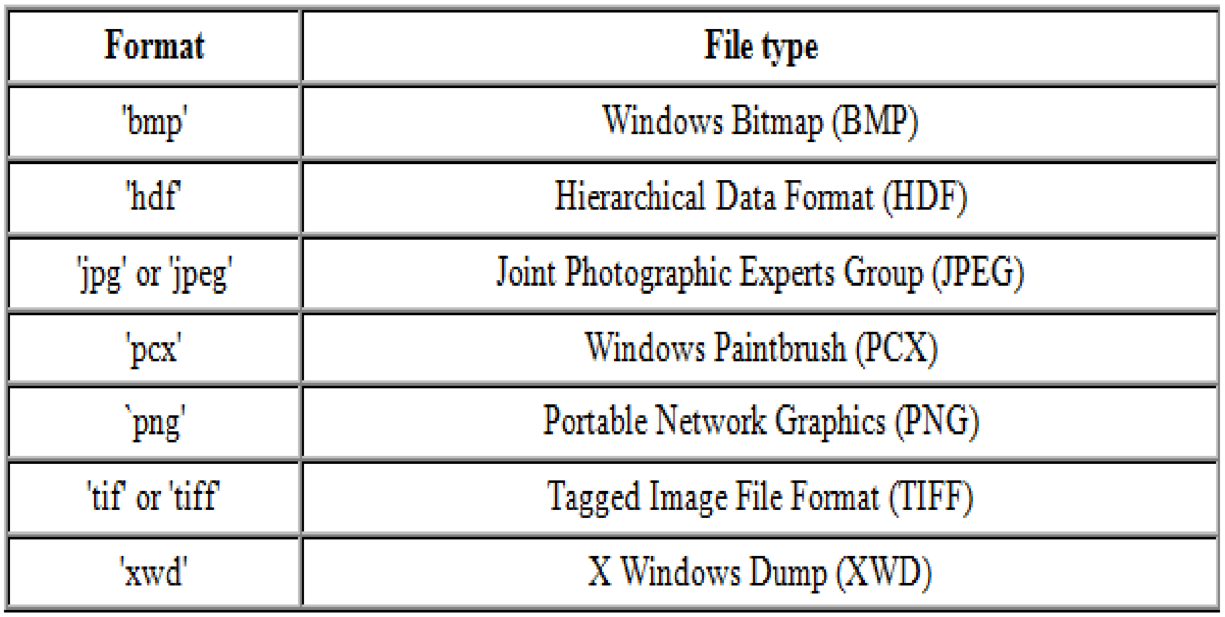
A = imread(filename,fmt)

Description:

A = imread(filename,fmt) reads a grayscale or truecolor image named filename into A. If the file contains a grayscale intensity image, A is a two-dimensional array. If the file contains a truecolor (RGB) image, A is a three-dimensional (m-by-n-by-3) array.

If imread cannot find a file named filename, it looks for a file named filename.fmt. If you do not specify a string for fmt, the toolbox will try to discern the format of the file by checking the file header.

This table 1 lists the possible values for fmt:



Format	File type
'bmp'	Windows Bitmap (BMP)
'hdf'	Hierarchical Data Format (HDF)
'jpg' or 'jpeg'	Joint Photographic Experts Group (JPEG)
'pcx'	Windows Paintbrush (PCX)
`png'	Portable Network Graphics (PNG)
'tif' or 'tiff'	Tagged Image File Format (TIFF)
'xwd'	X Windows Dump (XWD)

Table1: Image Format

1.8 Display Image:

imshow*:* Displays the image

Syntax

```
imshow(I)

imshow(I,[low high])

imshow(RGB)

imshow(BW)

imshow(X,map)

imshow(filename)

himage = imshow(...)

imshow(..., param1, val1, param2, val2,...)
```

Description

1) imshow(I) displays the grayscale image I.

2) imshow(I,[low high]) displays the grayscale image I, specifying the display range for I in [low high]. The value low (and any value less than low) displays as black; the value high (and any value greater than high) displays as white**.[7]** Values in between are displayed as intermediate shades of gray, using the default number of gray levels. If you use an empty matrix ([]) for [low high], imshow uses [min(I(:)) max(I(:))]; that is, the minimum value in I is displayed as black, and the maximum value is displayed as white.

3) imshow(RGB) displays the truecolor image RGB.

4) imshow(BW) displays the binary image BW. imshow displays pixels with the value 0 (zero) as black and pixels with the value 1 as white.

5) imshow(X,map) displays the indexed image X with the colormap map. A color map matrix may have any number of rows, but it must have exactly 3 columns. Each row is interpreted as a color, with the first element specifying the intensity of red light, the second green, and the third blue. Color intensity can be specified on the interval 0.0 to 1.0.

6) imshow(filename) displays the image stored in the graphics file filename. The file must contain an image that can be read by imread or dicomread. imshow calls imread or dicomread to read the image from the file, but does not store the image data in the MATLAB workspace. If the file contains multiple images, the first one will be displayed. The file must be in the current directory or on the MATLAB path.

Remarks

Imshow is the toolbox's fundamental image display function, optimizing figure, axes, and image object property settings for image display. imtool provides all the image display capabilities of imshow but also provides access to several other tools for navigating and exploring images, such as the Pixel Region tool, Image Information tool, and the Adjust Contrast tool. imtool presents an integrated environment for displaying images and performing some common image processing tasks.

Examples:

Display an image from a file.

X= imread('moon.tif');

imshow(X).

1.9 Image formats supported by Matlab:

A digital image is composed of pixels which can be thought of as small dots on the screen. A digital image is an instruction of how to color each pixel. We will see in detail later on how this is done in practice. A typical size of an image is 512-by-512 pixels. Later on in the course you will see that it is convenient to let the dimensions of the image to be a power of 2.

For example, 2^9=512. In the general case we say that an image is of size *m*-by-*n* if it is composed of *m* pixels in the vertical direction and *n* pixels in the horizontal direction.

Let us say that we have an image on the format 512-by-1024 pixels. This means that the data for the image must contain information about 524288 pixels, which requires a lot of memory! Hence, *c*ompressing images is essential for efficient image processing. You will later on see how Fourier analysis and Wavelet analysis can help us to compress an image significantly. There are also a few "computer scientific" tricks (for example entropy coding) to reduce the amount of data required to store an image.

The following image formats are supported by Matlab:

- BMP
- HDF
- JPEG
- PCX
- TIFF
- XWB

Most images you find on the Internet are JPEG-images which is the name for one of the most widely used compression standards for images. If you have stored an image you can usually see from the suffix what format it is stored in. For example, an image named myimage.jpg is stored in the JPEG format and we will see later on that we can load an image of this format into Matlab.

Working formats:

If an image is stored as a JPEG-image on your disc we first read it into Matlab. However, in order to start working with an image, for example perform a wavelet transform on the image, we must convert it into a different format. This section explains four common formats.

1. Intensity image (gray scale image):

This is the equivalent to a "gray scale image" and this is the image we will mostly work with in this course. It represents an image as a matrix where every element has a value corresponding to how bright/dark the pixel at the corresponding position should be colored.

There are two ways to represent the number that represents the brightness of the pixel: The double class (or data type). This assigns a floating number ("a number with decimals") between 0 and 1 to each pixel. The value 0 corresponds to black and the value 1 corresponds to white. The other class is called uint8 which assigns an integer between 0 and 255 to represent the brightness of a pixel.

The value 0 corresponds to black and 255 to white. The class uint8 only requires roughly 1/8 of the storage compared to the class double. On the other hand, many mathematical functions can only be applied to the double class.

2. Binary image:

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

3. Indexed image:

This is a practical way of representing color images. (In this course we will mostly work with gray scale images but once you have learned how to work with a gray scale image you will also know the principle how to work with color images.) An indexed image stores an image as two matrices.

The first matrix has the same size as the image and one number for each pixel. The second matrix is called the color map and its size may be different from the image. The numbers in the first matrix is an instruction of what number to use in the color map matrix.

4. RGB image

This is another format for color images. It represents an image with three matrices of sizes matching the image format. Each matrix corresponds to one of the colors red, green or blue and gives an instruction of how much of each of these colors a certain pixel should use.

5. Multiframe image

In some applications we want to study a sequence of images. This is very common in biological and medical imaging where you might study a sequence of slices of a cell. For these cases, the multi frame format is a convenient way of working with a sequence of images. In case you choose to work with biological imaging later on in this course, you may use this format.

1.10 Conversion between different formats:

The following table 2 shows how to convert between the different formats given above. All these commands require the Image processing tool box.

Image format conversion (Within the parenthesis type the name of the image you wish to convert.)	
Operation:	Matlab command:
Convert between intensity/indexed/RGB format to binary format.	dither()
Convert between intensity format to indexed format.	gray2ind()

Convert between indexed format to intensity format.	ind2gray()
Convert between indexed format to RGB format.	ind2rgb()
Convert a regular matrix to intensity format by scaling.	mat2gray()
Convert between RGB format to intensity format.	rgb2gray()
Convert between RGB format to indexed format.	rgb2ind()

Table2: Matlab commands

The command mat2gray is useful if you have a matrix representing an image but the values representing the gray scale range between, let's say, 0 and 1000. The command mat2gray automatically re scales all entries so that they fall within 0 and 255 (if you use the uint8 class) or 0 and 1 (if you use the double class).

How to read files

When you encounter an image you want to work with, it is usually in form of a file (for example, if you down load an image from the web, it is usually stored as a JPEG-file). Once we are done processing an image, These commands require the Image processing tool box! .The way of reading and writing the image files is shown in table3.

Reading and writing image files	
Operation:	Matlab command:

Read an image. (Within the parenthesis you type the name of the image file you wish to read. Put the file name within single quotes ' '.)	imread()
Write an image to a file. (As the first argument within the parenthesis you type the name of the image you have worked with. As a second argument within the parenthesis you type the name of the file and format that you want to write the image to. Put the file name within single quotes ' '.)	imwrite(,)

Table3: Image reading and writing commands

Loading and saving variables in Matlab:

This section explains how to load and save variables in Matlab. Once you have read a file, you probably convert it into an intensity image (a matrix) and work with this matrix. Once you are done you may want to save the matrix representing the image in order to continue to work with this matrix at another time. The Image saves and load commands are shown in table 4

Loading and saving variables	
Operation:	Matlab command:
Save the variable X	save X
Load the variable X	load X

Table4: Image save and load commands

1.11 How to display an image in Matlab:

Here are a couple of basic Matlab commands (do not require any tool box) for displaying an image. The image display commands are shown in table 5.

Displaying an image given on matrix form	
Operation:	Matlab command:
Display an image represented as the matrix X.	imagesc(X)
Adjust the brightness. s is a parameter such that -1<s<0 gives a darker image, 0<s<1 gives a brighter image.	brighten(s)
Change the colors to gray.	colormap(gray)

Table5: image display commands

Sometimes your image may not be displayed in gray scale even though you might have converted it into a gray scale image.**[7]** You can then use the command color map (gray) to "force" Matlab to use a gray scale when displaying an image.

If you are using Matlab with an Image processing tool box installed, recommended is to use the command imshow to display an image.

histeq: Enhance contrast using histogram equalization.

Syntax:

```
J = histeq(I, hgram)
J = histeq(I, n)
[J, T] = histeq(I,...)
```

```
newmap = histeq(X, map, hgram)
newmap = histeq(X, map)
[newmap, T] = histeq(X,...)
```

Description:

histeq enhances the contrast of images by transforming the values in an intensity image, or the values in the colormap of an indexed image, so that the histogram of the output image approximately matches a specified hist

ogram.

J = histeq(I, hgram) transforms the intensity image I so that the histogram of the output intensity image J with length(hgram) bins approximately matches hgram. The vector hgram should contain integer counts for equally spaced bins with intensity values in the appropriate range: [0, 1] for images of class double, [0, 255] for images of class uint8, and [0, 65535] for images of class uint16**.[6]** histeq automatically scales hgram so that sum(hgram)=prod(size(I)). The histogram of J will better match hgram when length(hgram) is much smaller than the number of discrete levels in I.

J = histeq(I, n) transforms the intensity image I, returning in J an intensity image with n discrete gray levels. A roughly equal number of pixels is mapped to each of the n levels in J, so that the histogram of J is approximately flat. (The histogram of J is flatter when n is much smaller than the number of discrete levels in I.)

The default value for n is 64.[J, T] = histeq(I,...) returns the grayscale transformation that maps gray levels in the image I to gray levels in J.

newmap = histeq(X, map, hgram) transforms the colormap associated with the indexed image X so that the histogram of the gray component of the indexed image (X,newmap) approximately matches hgram.**[7]** The histeq function returns the transformed colormap in newmap. length(hgram) must be the same as size(map,1).

newmap = histeq(X, map) transforms the values in the colormap so that the histogram of the gray component of the indexed image X is approximately flat. It returns the transformed colormap in newmap.

[newmap, T] = histeq(X,...) returns the grayscale transformation T that maps the gray component of map to the gray component of newmap.

Examples

Enhance the contrast of an intensity image using histogram equalization.

```
I = imread('tire.tif');

J = histeq(I);

imshow(I)

figure, imshow(J)
```

FigA.1 : Enhance intensity level of image

imhist**: Display histogram of image data**

Syntax:

```
imhist(I)
imhist(I, n)
imhist(X, map)
[counts,x] = imhist(...)
```

Description:

imhist(I) displays a histogram for the image I above a grayscale colorbar. The number of bins in the histogram is specified by the image type.**[6]** If I is a grayscale image, imhist uses a default value of 256 bins. If I is a binary image, imhist uses two bins.

imhist(I, n) displays a histogram where n specifies the number of bins used in the histogram. n also specifies the length of the colorbar. If I is a binary image, n can only have the value 2.

imhist(X, map) displays a histogram for the indexed image X.**[7]** This histogram shows the distribution of pixel values above a colorbar of the colormapmap. The colormap must be at least as long as the largest index in X. The histogram has one bin for each entry in the colormap.

[counts,x] = imhist(...) returns the histogram counts in counts and the bin locations in x so that stem(x,counts) shows the histogram. For indexed images, imhist returns the histogram counts for each colormap entry; the length of counts is the same as the length of the colormap.

Examples

```
I = imread('pout.tif');

imhist(I);

histeq;
```

Source Code:

```
Clc;
close all;
clear all;
i=imread('test1.JPG');
I=rgb2gray(i);
LEN=31;
THETA=11;
PSF=fspecial('motion',LEN,THETA);
blurred=imfilter(I,PSF,'circular','conv');
figure,imshow(blurred);
title('blurred image');
fspecial('gaussian',[5 5],0.5);
k = deconvwnr(blurred, PSF);
figure,imshow(k);
title('sharped image');
imagen=k;
% Convert to gray scale
if size(imagen,3)==3 %RGB image
    imagen=rgb2gray(imagen);
figure(2)
imshow(imagen)
end
% Convert to BW
threshold = graythresh(imagen);
imagen =~im2bw(imagen,threshold);
figure(3)
imshow(imagen)
% Remove all object containing fewer than 30 pixels
imagen = bwareaopen(imagen,2);
figure(4);imshow(imagen)
%Storage matrix word from image
word=[ ];
re=imagen;
%Opens text.txt as file for write
fid = fopen('text.txt', 'wt');
```

```
load templates
global templates
% Compute the number of letters in template file
num_letras=size(templates,2);
while 1
    %Fcn 'lines' separate lines in text
    [fl re]=lines(re);
    imgn=fl;
    figure(5)
    imshow(fl);pause(0.5)
    % Label and count connected components
    [L Ne] = bwlabel(imgn);
    for n=1:Ne
        [r,c] = find(L==n);
        % Extract letter
        n1=imgn(min(r):max(r),min(c):max(c));
        % Resize letter (same size of template)
        img_r=imresize(n1,[42 24]);
         figure(6);imshow(img_r);pause(0.5)
        %-------------------------------------------------------------------
        % Call fcn to convert image to text
        letter=read_letter(img_r,num_letras);
        % Letter concatenation
        word=[word letter];
    end
        fprintf(fid,'%s\n',word);%Write 'word' in text file (upper)
    % Clear 'word' variable
    word=[ ];
    %*When the sentences finish, breaks the loop
    if isempty(re)  %See variable 're' in Fcn 'lines'
        break
    end
end
fclose(fid);
%Open 'text.txt' file
winopen('text.txt')
```

```
%CREATE TEMPLATES
%Letter
A=imread('letters_numbers\A.bmp');B=imread('letters_numbers\B.bmp');
C=imread('letters_numbers\C.bmp');D=imread('letters_numbers\D.bmp');
E=imread('letters_numbers\E.bmp');F=imread('letters_numbers\F.bmp');
G=imread('letters_numbers\G.bmp');H=imread('letters_numbers\H.bmp');
I=imread('letters_numbers\I.bmp');J=imread('letters_numbers\J.bmp');
K=imread('letters_numbers\K.bmp');L=imread('letters_numbers\L.bmp');
M=imread('letters_numbers\M.bmp');N=imread('letters_numbers\N.bmp');
O=imread('letters_numbers\O.bmp');P=imread('letters_numbers\P.bmp');
Q=imread('letters_numbers\Q.bmp');R=imread('letters_numbers\R.bmp');
S=imread('letters_numbers\S.bmp');T=imread('letters_numbers\T.bmp');
U=imread('letters_numbers\U.bmp');V=imread('letters_numbers\V.bmp');
W=imread('letters_numbers\W.bmp');X=imread('letters_numbers\X.bmp');
Y=imread('letters_numbers\Y.bmp');Z=imread('letters_numbers\Z.bmp');
%Number
one=imread('letters_numbers\1.bmp');  two=imread('letters_numbers\2.bmp');
three=imread('letters_numbers\3.bmp');four=imread('letters_numbers\4.bmp');
five=imread('letters_numbers\5.bmp'); six=imread('letters_numbers\6.bmp');
seven=imread('letters_numbers\7.bmp');eight=imread('letters_numbers\8.bmp');
nine=imread('letters_numbers\9.bmp'); zero=imread('letters_numbers\0.bmp');
%*-*-*-*-*-*-*-*-*-*-*-
letter=[A B C D E F G H I J K L M...
    N O P Q R S T U V W X Y Z];
number=[one two three four five...
    six seven eight nine zero];
character=[letter number];
templates=mat2cell(character,42,[24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 24]);
save ('templates','templates')
clear all
```

```
-%READ LETTER FUNCTION
function letter=read_letter(imagn,num_letras)
% Computes the correlation between template and input image
global templates
comp=[ ];
for n=1:num_letras
    sem=corr2(templates{1,n},imagn);
    comp=[comp sem];
end
vd=find(comp==max(comp));
%*-*-*-*-*-*-*-*-*-*-*-*-*-
if vd==1
    letter='A';
elseif vd==2
    letter='S';
elseif vd==3
    letter='C';
elseif vd==4
    letter='D';
elseif vd==5
    letter='E';
elseif vd==6
    letter='F';
elseif vd==7
    letter='G';
elseif vd==8
    letter='H';
elseif vd==9
    letter='I';
elseif vd==10
    letter='J';
elseif vd==11
    letter='K';
elseif vd==12
    letter='L';
elseif vd==13
    letter='M';
```

```
elseif vd==14
    letter='N';
elseif vd==15
    letter='O';
elseif vd==16
    letter='P';
elseif vd==17
    letter='Q';
elseif vd==18
    letter='R';
elseif vd==19
    letter='S';
elseif vd==20
    letter='T';
elseif vd==21
    letter='U';
elseif vd==22
    letter='V';
elseif vd==23
    letter='W';
elseif vd==24
    letter='X';
elseif vd==25
    letter='Y';
elseif vd==26
    letter='Z';
    %*-*-*-*-*
elseif vd==27
    letter='1';
elseif vd==28
    letter='2';
elseif vd==29
    letter='3';
elseif vd==30
    letter='4';
elseif vd==31
    letter='5';
```

```
elseif vd==32
    letter='6';
elseif vd==33
    letter='7';
elseif vd==34
    letter='8';
elseif vd==35
    letter='9';
else
    letter='I';
end

% LINES FUNCTION
function [fl re]=lines(im_texto)
 im_texto=clip(im_texto);
num_filas=size(im_texto,1);
for s=1:num_filas
    if sum(im_texto(s,:))==0
        nm=im_texto(1:s-1, :); % First line matrix
        rm=im_texto(s:end, :);% Remain line matrix
        fl = clip(nm);
        re=clip(rm);
              subplot(2,1,1);imshow(fl);
              subplot(2,1,2);imshow(re);
        break
    else
        fl=im_texto;%Only one line.
        re=[ ];
    end
end
 function img_out=clip(img_in)
[f c]=find(img_in);
img_out=img_in(min(f):max(f),min(c):max(c));%Crops image
```