Chapter 1

Introduction and Motivation

Optical character recognition, usually abbreviated to OCR, is the mechanical or electronic translation of images of printed text (usually captured by a scanner) into machine-editable text [48]. History of building such devices dates back to time before the invention of digital computers.

Use of various mathematical techniques in the subtasks of OCR technology is also equally old. In this chapter we briefly introduce various steps in design and development of an OCR system. This is followed by basic terminology used in the subsequent chapters. A detailed analysis of Gujarati script from OCR perspective is also presented with motivation to choose/develop various mathematical techniques for analysis and recognition of a Gujarati document.

1.1 Introduction

Optical character recognition, in the context of computers, means converting an image of a machine printed document in to a computer file which can be stored, edited, indexed and searched more efficiently than the image of the document itself. It is expected from a "usable" OCR system that it can take image in one of the popular image formats like Bitmap (BMP), Joint Photographic Experts Group (JPEG), Graphics Interchange Format(GIF), Tagged Image File Format (TIFF), Portable Network Graphics (PNG) etc. and can generate a computer readable file in format supported by one of the popular text editors(plain text file, .TXT), word processors(Microsoft Word - .DOC or

Open Office - .ODF / .ODT) or other document viewers (Acrobat Reader - PDF, Bowsers - HTML) etc., keeping the format of the document (positioning and style of text and non-text) intact (Fig. 1.1).

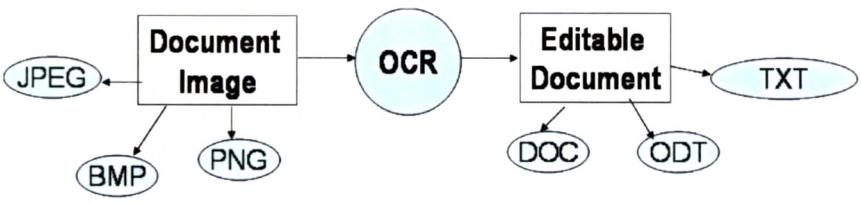


Figure 1.1: Goal of an OCR system

Various subtasks of an OCR system can be broadly divided into four groups (Fig. 1.2):

1. Image Acquisition and Preprocessing
2. Segmentation
3. Recognition
4. Document Reconstruction / Layout Restoration

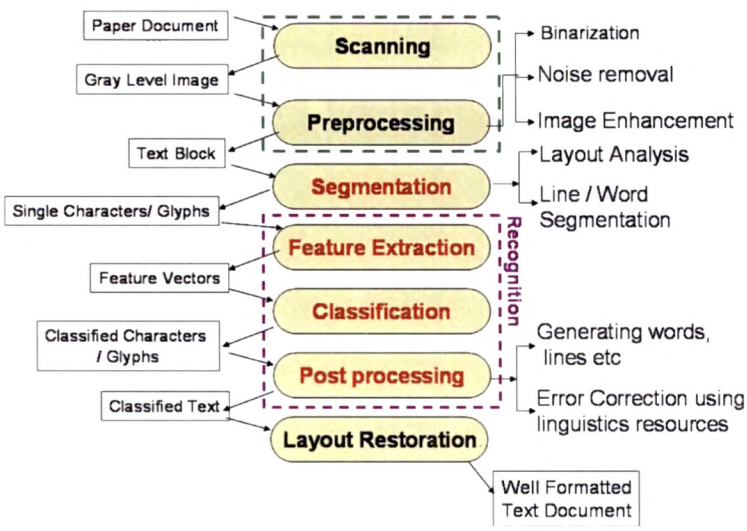


Figure 1.2: Subtasks of OCR system

The selection of mathematical techniques for performing first and the last task do not depend on the language / script of the text in the document where as recognition process is highly script dependent. Out of the processes used for segmentation a few are script/language dependent and the others are more or

less script / language independent.

As a part of this work we have explored use of various mathematical techniques to carry out the script dependent tasks at various stages of Gujarati OCR system. Following is a brief description of each of these tasks. Many of these processes are discussed in details in the subsequent sections with the algorithm and mathematics behind them.

1. **Image Acquisition and Preprocessing**
   During this process the digital image of the document under consideration is acquired and the image is processed to facilitate the subsequent processing. In most of the cases it undergoes processing like *Binarization, Noise Removal and Contrast Enhancement.*

2. **Segmentation**
   As the name suggests here we try to segment the text blocks in various elements like line, word, character/glyph etc. A detailed discussion about the mathematical techniques used for segmentation is given in next chapter. Ideally at the end of all stages of this tasks we have a set of symbols, technically referred to as *glyphs*, that needs to be recognized. Glyph

3. **Recognition**
   Segmented glyphs are recognized here. The decision about the unit of recognition will drive the segmentation process, the depth of the segmentation to be precise. Recognition can be divided into three subtasks viz. *Feature Extraction, Classification and Post Processing.* A separate chapter is devoted to each of these tasks to describe the concept in general and to discuss techniques used for Gujarati script.

4. **Document Reconstruction / Layout Restoration**
   The output of the previous task is text blocks which corresponds to their images extracted during the layout analysis. During this step those text blocks are arranged appropriately to restore the layout of the printed document in the soft copy, using the information acquired at the layout analysis subtask of the first step. At the end of this process the OCR system is expected to produce editable copy of the document image scanned at the first step keeping the formatting intact.

1.2 History Of OCR

Attempts for the construction of OCR for English and other Latin based scripts can be traced back up to about a hundred years i.e. even before the

advent of digital computers. In 1929 Gustav Tauschek obtained a patent on OCR in Germany, followed by Handel who obtained a US patent on OCR in USA in 1933. In 1935 Tauschek was also granted a US patent on his method (U.S. Patent 2,026,329). Tauschek's machine was a mechanical device that used templates. A photo detector (photoelectric cell) was placed so that when the template and the character to be recognized were lined up for an exact match and a light was directed towards them, no light would reach the photo detector [48].

These attempts started even before the advent of digital computers and has been consistently pursued since then. Due to these efforts the current state-of-art OCR for these European scripts could achieve an accuracy of 99% and more. In contrast, the development of OCR systems for Indic scripts started much later and is yet in the development stage.

Several commercial OCR systems are available for various European and other scripts of the world. ABBYY FineReader, Readiris, OmniPage are a few popular commercial OCR products which cater to different scripts of the world.

1.3 Image Acquisition [19]

Image acquisition, as the name suggests, is a process of creating an image (Digital in our case) of an object (Document for this discussion) through some imaging device like digital camera, flat bed scanner, book scanners, CT Scanning or MRI devices etc.

The principal manner in which array of sensors are used in the process of image acquisition is shown in Fig. 1.3. In this case energy from an illumination source being reflected from a scene element, but, in some cases, the energy also could be transmitted through the scene elements. The first function performed by the imaging system shown is to collect the incoming energy and focus it onto an image plane. If the illumination is light, the front end of the imaging system is a lens, which projects the viewed scene onto the lens focal plane, referred as image plane in this image. The sensor array, which is coincident with the focal plane, produces outputs proportional to the integral of the light received at each sensor. Digital and analog circuitry sweep these outputs and convert them to a video signal, which is then digitized by another section of the imaging system. The output is a digital image.

Mathematically, an image may be defined as a two-dimensional function,

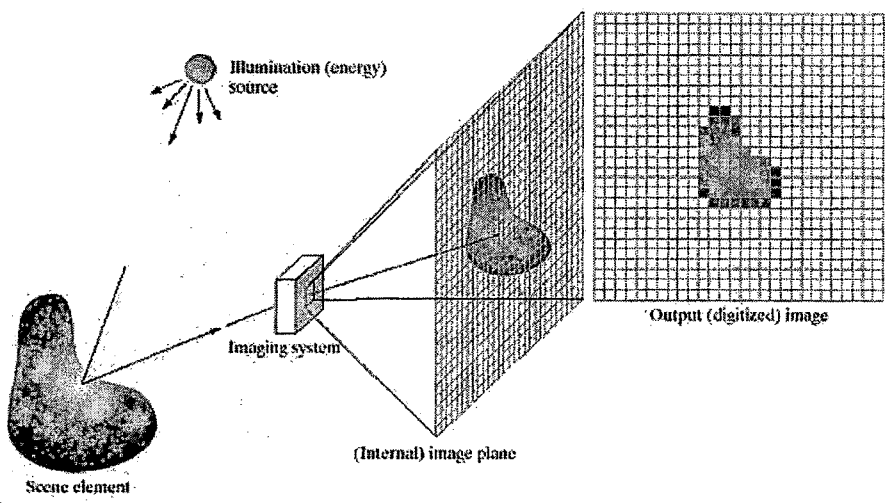


Figure 1.3: Example of Image Acquisition

$f(x,y)$, where x and y are spatial (plane) coordinates, and the amplitude of f at any pair of coordinates (x,y) is called the intensity or gray level of the image at that point.When x, y, and the amplitude values of f are all finite, discrete quantities, we call the image a *digital image*. Note that a digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as *picture elements*, *image elements*, *pels*, and *pixels*.

Digital Image

Pixel

In the case of OCR we use light as source of illumination. Light that is void of color is called *achromatic* or *monochromatic light*. The only attribute of such light is its intensity, or amount. The term *gray level* generally is used to describe monochromatic intensity because it ranges from black, to grays, and finally to white.

Gray Level

As mentioned above, we shall denote images by two-dimensional functions of the form $f(x,y)$. The value or amplitude of f at spatial coordinates (x,y) is a positive scalar quantity whose physical meaning is determined by the source of the image. When an image is generated from a physical process, the function values at various points (x,y) are proportional to energy radiated by a physical source (e.g. electromagnetic waves). As a consequence, $f(x,y)$ must be nonzero and finite; that is,

$$0 < f(x,y) < \infty \tag{1.1}$$

The function $f(x,y)$ may be characterized by two components: (1) the amount

of source illumination incident on the scene being viewed, and (2) the amount of illumination reflected by the objects in the scene.Appropriately, these are called the illumination and reflectance components and are denoted by $i(x, y)$ and $r(x, y)$, respectively. The two functions combine as a product to form $f(x, y)$:

$$f(x, y) = i(x, y)r(x, y) \tag{1.2}$$

where

$$0 < i(x, y) < \infty \tag{1.3}$$

and

$$0 < r(x, y) < 1 \tag{1.4}$$

The nature of $i(x, y)$ is determined by the illumination source, and $r(x, y)$ is determined by the characteristics of the imaged objects. The bounds in Eqs. 1.3 and 1.4 are theoretical bounds and in practice these bounds may vary depending on the source of illumination and the object being imaged respectively.

We call the intensity of a monochrome image at any coordinates (x, y) the *gray level(ℓ)* of the image at that point.That is,

$$\ell = f(x, y) \tag{1.5}$$

From Eqs. 1.2 through 1.4, it is evident that ℓ lies in the range

$$L_{min} \leq \ell \leq L_{max} \tag{1.6}$$

Theoretically, L_{min} should be positive ($L_{min} \geq 0$) and L_{max} be finit($L_{max} < \infty$).

The interval $[L_{min}, L_{max}]$ is called the *gray scale*. Common practice is to shift this interval numerically to the interval $[0, L - 1]$, where $\ell = 0$ is considered black and $\ell = L - 1$ is considered white on the gray scale.All intermediate values are shades of gray varying from black to white. Gray Scale

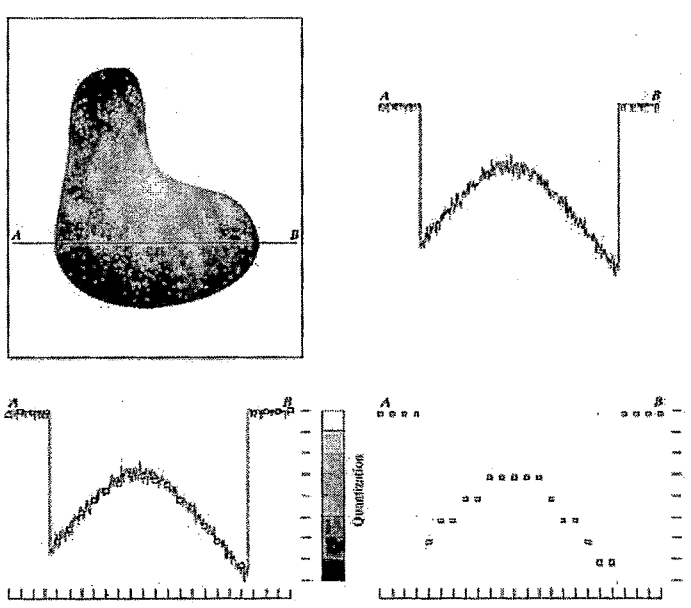
It is clear from the above discussion that the output of most sensors is a continuous voltage waveform whose amplitude and spatial behavior are related to the physical phenomenon being sensed. Therefore, to create a digital image, we need to convert the continuous sensed data into digital form. This involves two processes: *sampling* and *quantization*.

1.3.1 Sampling and Quantization

The basic idea behind sampling and quantization is illustrated in Fig. 1.4. Figure 1.4(a) shows a continuous image, $f(x, y)$, that we want to convert to digital form. An image may be continuous with respect to the x- and y-coordinates, and also in amplitude. To convert it to digital form, we have to sample the function in both coordinates and in amplitude. Digitizing the coordinate values is called *sampling*. Digitizing the amplitude values is called *quantization*. The one-dimensional function shown in Fig. 1.4(b) is a plot of

Sampling

Quantization



a	b
c	d

Figure 1.4: Image Digitization : (a) Continuous image. (b) A scan line from A to B in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line

amplitude (gray level) values of the continuous image along the line segment AB in Fig. 1.4(a). The random variations are due to image noise. To sample this function, we take equally spaced samples along line AB, as shown in Fig. 1.4(c). The location of each sample is given by a vertical tick mark in the bottom part of the figure.The samples are shown as small white squares superimposed on the function. The set of these discrete locations gives the sampled function. However, the values of the samples still span (vertically) a continuous range of gray-level values. In order to form a digital function, the gray-level values also must be converted (*quantized*) into discrete quantities. The right side of Fig. 1.4(c) shows the gray-level scale divided into eight dis-

crete levels, ranging from black to white. The vertical tick marks indicate the specific value assigned to each of the eight gray levels. The continuous gray levels are quantized simply by assigning one of the eight discrete gray levels to each sample. The assignment is made depending on the vertical proximity of a sample to a vertical tick mark. The digital samples resulting from both sampling and quantization are shown in Fig. 1.4(d). Starting at the top of the image and carrying out this procedure line by line produces a two-dimensional digital image.

Sampling in the manner just described assumes that we have a continuous image in both coordinate directions as well as in amplitude. In practice, the method of sampling is determined by the sensor arrangement used to generate the image. When an image is generated by a single sensing element combined with mechanical motion, as in flat bed scanners, the output of the sensor is quantized in the manner described above. However, sampling is accomplished by selecting the number of individual mechanical increments at which we activate the sensor to collect data. Mechanical motion can be made very exact so, in principle, there is almost no limit as to how fine we can sample an image. However, practical limits are established by imperfections in the optics used to focus on the sensor an illumination spot that is inconsistent with the fine resolution achievable with mechanical displacements.

The quality of a digital image is determined to a large degree by the number of samples and discrete gray levels used in sampling and quantization.

1.3.2 Digital Images: Representation and Storage

The result of sampling and quantization is a matrix of real numbers. We will use two principal ways to represent digital images. Assume that an image $f(x, y)$ is sampled so that the resulting digital image has M rows and N columns. The values of the coordinates (x, y) now become discrete quantities. For notational clarity and convenience, we shall use integer values for these discrete coordinates. Thus, the values of the coordinates at the origin are $(x, y) = (0, 0)$. The next coordinate values along the first row of the image are represented as $(x, y) = (0, 1)$. It is important to keep in mind that the notation $(0, 1)$ is used to signify the second sample along the first row. It does not mean that these are the actual values of physical coordinates when the image was sampled. Figure 1.5 shows the coordinate convention used for this work.

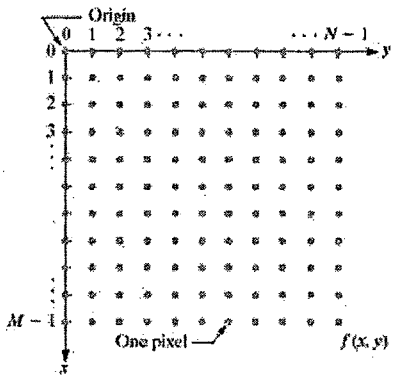


Figure 1.5: Image Coordinate System

Hence, a digital image of dimension $M \times N$ can be written in a matrix form as follows:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{1.7}$$

For the sake of convenience, the image matrix in eq. 1.7 is written in more common matrix notation :

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix} \tag{1.8}$$

Clearly, $a_{ij} = f(i,j)$ and hence representations in eqs. 1.7 and 1.8 are identical. Also each element of this image corresponds to a color / gray level of a pixel in the corresponding digital image.

The amount of storage occupied by any image on a storage device depends on number of bits used to store gray level / color information of each pixel of the image. For example, if k bits are used to store one pixel then image with M pixel rows and N pixel columns will take $M \times N \times k$ bits of storage space. The number of bits used to store gray level for each pixel will decide value for L_{max}in eq. 1.6 and hence of $L-1$ in the shifted scale. i.e. for image where k bits are used to store value of gray level of each pixel, number of possible gray level are

$$L = 2^k \tag{1.9}$$

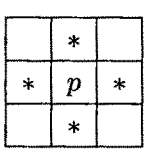
For this work we have scanned the pages with text keeping the value of $k = 8$ i.e with total 256 possible gray levels (values ranging from 0 through 255) for each pixels. The range of values spanned by the gray scale is called the *dynamic range* of an image.

Dynamic Range

A pixel p at coordinates (x, y) has four *horizontal* and *vertical neighbors* whose coordinates are given by

Neighbors of a Pixel

$$(x+1, y),\ (x-1, y),\ (x, y+1),\ (x, y-1)$$



This set of pixels, called the *4-neighbors* of p, is denoted by $N_4(p)$. Each pixel is a unit distance from (x, y), and some of the neighbors of p may lie outside the digital image if (x, y) is on the border of the image.

The four diagonal neighbors of p have coordinates

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$$

and are denoted by $N_D(p)$. These points, together with the 4-neighbors, are called the *8-neighbors* of p, denoted by $N_8(p)$. As before, some of the points in $N_D(p)$ and $N_8(p)$ fall outside the image if (x, y) is on the border of the image.

Digital image acquired like this is likely to be far from the quality needed for OCR to work properly. This may be because of artifacts due to different sources like quality of the paper, printing technology, ink, dust on the glass of the scanner, problem in digitization etc., other qualitative defects like poor contrast or unwanted break(merge) of the characters. If such an image is subject to the subsequent processes then it is likely to result into unexpected segmentation. Hence, the recognition will also suffer due to this. Therefore, before proceeding further those artifacts need to be removed and hence the images are to be preprocessed. Further, a typical document, say a page from a magazine or a book, is likely to have many other objects in addition to the text like images, graphics, lines etc.. Hence, it is also required to analyze document image for presence of any of these *non-text* objects in the image and mark the areas where the *text* to be recognized is present.

Following section touches the preprocessing techniques used for removing ar-

tifacts and the mathematics used in the process.

1.4 Preprocessing

The document image generated as described in the previous section may be subjected to various operations to remove the artifacts and enhance the image quality so that subsequent processes give expected results. The operations can be divided in two groups : *Linear* and *Nonlinear Operations.*

Let H be an operator whose input and output are images. H is said to be a *linear operator* if, for any two images f and g and any two scalars a and b,

$$H(af + bg) = aH(f) + bH(g) \tag{1.10}$$

In other words, the result of applying a linear operator to the sum of two images that have been multiplied by the constants shown, is identical to applying the operator to the images individually, multiplying the results by the appropriate constants, and then adding those results. In order that these operations to be feasible, the two images must be sampled at the same level and algebraic operation must be defined among different level of quantization. Thus any linear operator can be regarded linear operator on the Linear space of $M \times N$ Matrices. For example, an operator whose function is to compute the sum of K images is a linear operator. An operator that computes the absolute value of the difference of two images is not. An operator that fails the test of Eq. (1.10) is by definition nonlinear. Linear operations are exceptionally important in image processing because they are based on a significant body of well-understood theoretical and practical results.Although nonlinear operations sometimes offer better performance, they are not always predictable, and for the most part are not well understood theoretically.

Depending on the quality of the document image it may need to be preprocessed by one or more of the following preprocessing algorithm.

1.4.1 Geometric Spatial Transformations

Geometric transformations modify the spatial relationship between pixels in an image. These transformations often are called *rubber-sheet* transformations because they may be viewed as analogous to "printing" an image on a sheet of rubber and then stretching the sheet according to the predefined set of rules. In terms of digital image processing, a geometric transformation consists of

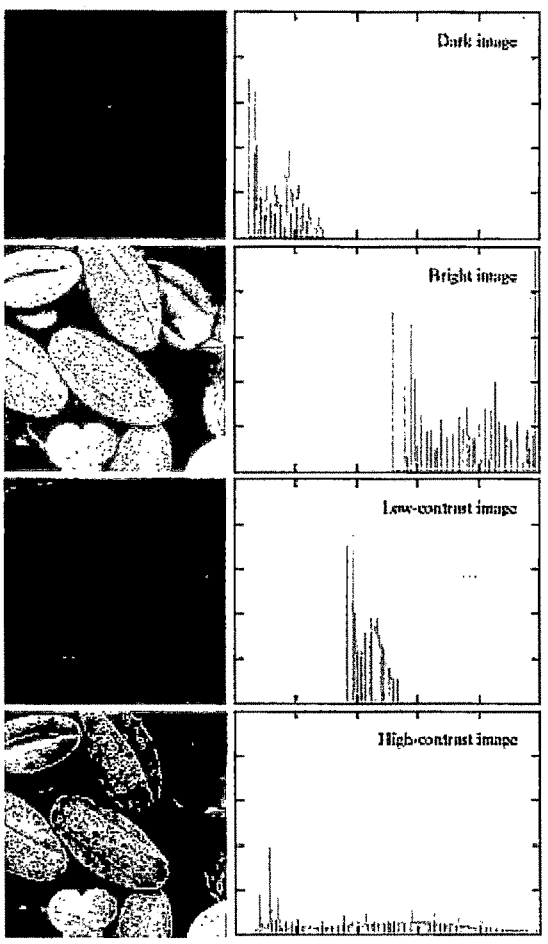


Figure 1.6: Image Quality and Histogram

two basic operations:(1) a spatial transformation of coordinates and (2) intensity interpolation that assigns intensity to the spatially transformed pixels.

The transformation of coordinates may be expressed as

$$(x, y) = T(v, w) \tag{1.11}$$

where (v, w) are pixel coordinates in the original image and (x, y) are the corresponding pixel coordinates in the transformed image. For example, the transformation $(x, y) = T(v, w) = (v/2, w/2)$ shrinks the originalimage to half its size in both spatial directions. The general form for these operations is

$$[x \quad y \quad 1] = [v \quad w \quad 1]\mathbf{T} = [v \quad w \quad 1] \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix} \tag{1.12}$$

This transformation can scale, rotate, translate or shear a set of coordinate points, depending on the value chosen for the elements of matrix **T**. The matrix values used for the common operations in OCR : scaling and rotation are given in eq. 1.13 and 1.14 respectively.

$$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.13}$$

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.14}$$

1.4.2 Contrast Enhancement

Contrast for a gray scale image is defined as the difference of gray scale between two neighboring pixels. The *histogram*[19] of a digital image with gray levels in the range $[0, L-1]$ is a discrete function

Contrast

Histogram

$$h(r_k) = n_k \tag{1.15}$$

where r_k is the k^{th} gray level and n_k is the number of pixels in the image having gray level r_k. Analysis of histogram can reveal important information about contrast in the image. Figure 1.6 illustrates this.

One of the most common defects found in a recorded image is its poor contrast. This degradation may be caused by inadequate lighting, aperture size and / or shutter speed. In document images this could be due to deterioration in the page color due to aging or the actual color of the paper. As shown in Fig. 1.6, such defect is reflected in the histogram of the image. In this case, the contrast can be improved by scaling the gray level of each pixel, so that image gray level occupy entire dynamic range available. Due to effect of such scaling, the Histogram of the image gets stretched and hence such scaling is called *histogram stretching*. Let T be the transformation function responsible for scaling gray level m to l after stretching. That is

Histogram Stretching

$$l = T(m) \tag{1.16}$$

T must satisfy the following conditions [9] :

1. $T(m)$ must be monotonically increasing in the interval $[m_{min}, m_{max}]$ i.e.

$$m_1 < m_2 \Rightarrow l_1 = T(m_1) \leq l_2 = T(m_2) \tag{1.17}$$

   That means that transformed gray level l must preserve the order from black to white.

2. $l_{min} \leq l \leq l_{max}$ i.e. transformed gray level must lie within the allowed range of gray level.

Where $[l_{min}, l_{max}]$ and $[m_{min}, m_{max}]$ are the available gray level range and gray level range in the given image, respectively.

The transformation function described above could be a straight line having slope $(l_{max} - l_{min})/(m_{max} - m_{min})$ [9] i.e.

$$l = T(m) = \frac{l_{max} - l_{min}}{m_{max} - m_{min}}(m - m_{min}) + l_{min} \tag{1.18}$$

This type of the stretching is referred to as *linear stretching.* Linear Stretching

The slope of the transformation function is always greater than or equal to 1. If the slope is 1 there is no enhancement. In fact, image is enhanced only when the slope is much greater than 1. Sometimes only a portion of the image gray level range of the input image is strongly stretched to emphasize certain features of the image [9]. For example, the transformation defined over $[l_{min}, l_{max}]$

$$l = \begin{cases} \frac{[3a - b - 2\,l_{min}]m + [b - a]l_{min}}{2\,[a - l_{min}]} & for\ l_{min} \leq m < a \\ \frac{4m - a - b}{2} & for\ a \leq m \leq b \\ \frac{[2\,l_{max} + a - 3b]\,m + [b - a]l_{max}}{2\,[l_{max} - b]} & for\ b < m \leq l_{max} \end{cases} \tag{1.19}$$

stretches the subrange$[a,\ b]$ by a factor of 2. The remaining portion of the entire gray level range, i.e. $[l_{min}, a)$ and $(b,\ l_{max}]$ have been compressed to satisfy the condition 2. This is known as *piecewise stretching.* It should also be noted that the rate of stretching changes abruptly at $m = a$ and at $m = b$. Smoother changes may be achieved by employing non-linear transformation function. Piecewise Stretching

1.4.3 Binarization

It may be recalled that the document image produced through scanning process is a gray scale image and hence due to quality of the paper used for the

printing the background of page need not produce perfect white color in the scanning and it can have varying shades of gray. Further, the text printed on the back side can also produce noise in terms of unwanted gray pixels.

Binarization
Thresholding

Binarization or *thresholding* is the operation performed upon gray-scale document images which contain text or graphics [24]. The objective of binarization is to automatically choose a threshold that separates the foreground and background information. The output of the thresholding operation is a binary image whose one state will indicate the foreground objects, that is, printed text, a legend, a target, defective part of a material etc. while the complementary state will correspond to the background. Depending on the application, the foreground can be represented by gray-level 0, that is, black as for text, and the background by the highest luminance for document paper, that is 255 in 8-bit images, or conversely the foreground by white and the background by black [35]. Selection of an appropriate threshold is often a trial and error process.

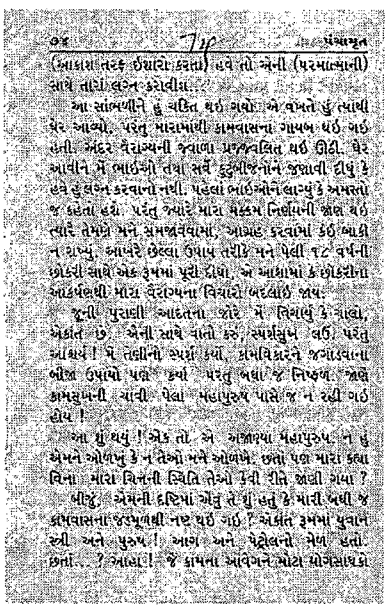
Mathematically, let I_b be the binarized version of gray scale image I with t being the threshold then,

$$I_b(x,y) = \begin{cases} 0 & \text{if } I(x,y) \leq t \\ 255 & \text{if } I(x,y) > t \end{cases} \tag{1.20}$$

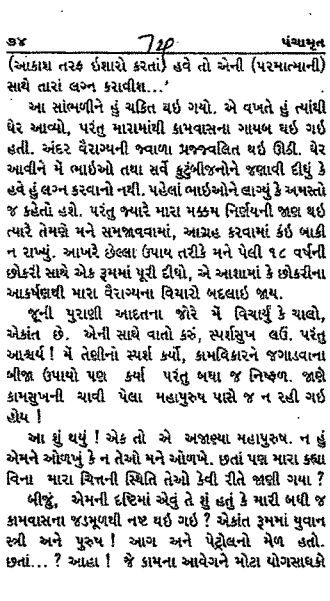
Figure 1.7(a) shows a scanned page of a Gujarati book. Background noise and the gray shade of the page is clearly visible there. Figure 1.7(b)is the binarized version of this image.

Selection of threshold is very important in the binarization process as selection of threshold higher than the ideal may produce broken characters and the threshold less than the ideal may produce touching characters. There are several ways in which the binarization process can be modeled.

Mathematically, if we consider the gray level histogram of a document image, which is a bimodal graph , ideally the threshold t should be at the bottom of the trough between the two peaks (Fig. 1.8(b)). In [35] Sezgin and Sankur have surveyed various binarization methods where as [41] documents evaluation of various binarization algorithms for document images. The thresholding methods can be categorized in six groups according to the information they are exploiting [35].



(a) Gujarati Document



(b) Result of Binarization

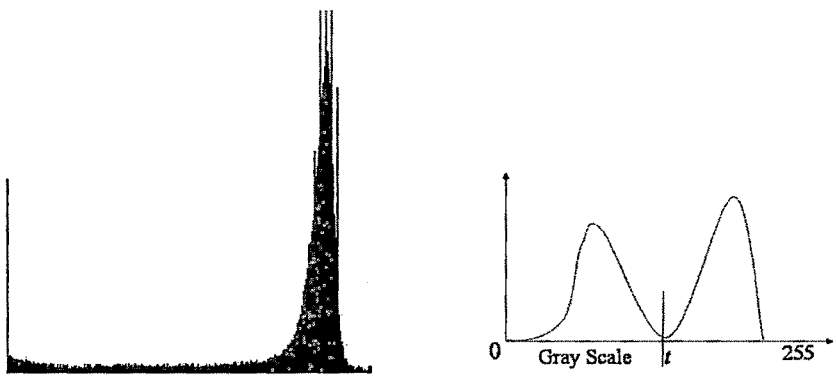
Figure 1.7: Binarization

These categories are:

1. **Histogram Shape-Based Thresholding Methods:**
   This category of methods achieves thresholding based on the shape properties of the histogram. Some of the algorithms use the shape properties like, the distance from the convex hull of the histogram, a smoothed two-peaked representation of the histogram, a more crude rectangular approximation to the lobes of the histogram where as some other algorithms search explicitly for peaks and valleys, or implicitly for overlapping peaks via curvature analysis.

2. **Clustering-Based Thresholding Methods :**
   In this class of algorithms, the grey-level data is grouped into two clusters corresponding to the two lobes of a histogram (assumed distinct). Some algorithms are based on the midpoint of the peaks, some are based on the fitting of the mixture of Gaussians, some are based on Mean-square clustering while some use fuzzy clustering ideas.

3. **Entropy-Based Thresholding Methods :**
   This class of algorithms exploits the entropy of the distribution of the grey levels in a scene. The maximization of the entropy of the thresholded image is interpreted as indicative of maximum information transfer. Other authors try to minimize the cross-entropy between the input



(a) Gray scale Histogram of Document in Fig. 1.7(a)

(b) Threshold Selection

Figure 1.8: Gray Level Histogram

gray-level image and the output binary image as indicative of preservation of information, or a measure of fuzzy entropy and some authors used Shannon entropy-based thresholding.

4. **Thresholding Based on Attribute Similarity :**
   These algorithms select the threshold value based on some attribute quality or similarity measure between the original image and the binarized version of the image. These attributes can take the form of edge matching, shape compactness, grey-level moments, connectivity, texture, or ability of segmented objects. Some other algorithms evaluate directly the resemblance of the original grey-level image to binary image resemblance using fuzzy measure or resemblance of the cumulative probability distributions, or in terms of the quantity of information revealed as a result of segmentation.

5. **Spatial Thresholding Methods :**
   This class of algorithms utilizes not only grey value distribution but also dependency of pixels in a neighborhood, for example, in the form of context probabilities, correlation functions, co occurrence probabilities, local linear dependence models of pixels, 2-D entropy, etc.

6. **Locally Adaptive Thresholding :**
   In this class of algorithms, a threshold is calculated at each pixel, which depends on some local statistics like range, variance, or surface-fitting parameters of the pixel neighborhood.

In [35] exhaustive list of various thresholding methods in all the above categories is given. We discuss here only three methods, which we have used in this work. One of these methods proposed by Otsu, belongs to category 2 and

other proposed by Niblack and Sauvola & Pietaksinen, belong to category 6.

We introduce some more concepts discussed in [35] to explain these methods. Let $h(g)$ and $p(g)$ be histogram and the probability mass function (PMF) of the image respectively. Here $g = 0...G$ where G is the maximum luminance value in the image, typically 255 if 8-bit quantization is assumed. The cumulative probability function is defined as

$$P(g) = \sum_{i=0}^{g} p(i) \tag{1.21}$$

It is assumed that the PMF is estimated from the histogram of the image by normalizing it to the total number of samples. In the context of document processing, the foreground (object) becomes the set of pixels with luminance values less than T, while the background pixels have luminance value above this threshold. The foreground (object) and background PMFs are expressed as $P_f(g)$, $0 < g < T$ and $P_b(g)$, $T + 1 \leq g \leq G$, respectively, where T is the threshold value. The foreground and background area probabilities are calculated as:

$$P_f(T) = P_f = \sum_{g=0}^{T} p(g), \text{ and } P_b(T) = P_b = \sum_{g=T+1}^{G} p(g) \tag{1.22}$$

Otsu's method selects the global threshold

$$T = \arg\max \left\{ \frac{P(T)[1 - P(T)][m_f(T) - m_b(T)]^2}{P(T)\sigma_f^2(T) + [1 - P(T)]\sigma_b^2(T)} \right\} \tag{1.23}$$

Where,

$$m_f(T) = \sum_{g=0}^{T} g\, P(g) \text{ and } m_b(T) = \sum_{g=T+1}^{G} g\, P(g) \tag{1.24}$$

and

$$\sigma_f^2(T) = \sum_{g=1}^{T} [g - m_f(T)]^2 P(g) \text{ and } \sigma_b^2(T) = \sum_{g=T+1}^{G} [g - m_b(T)]^2 P(g). \tag{1.25}$$

Niblack's method [35] for binarization, which follows local adaptive thresholding, adapts the threshold according to the local mean $m(i, j)$ and standard deviation $s(i, j)$ of grey level values of the image f calculated on window of size $b \times b$ with pixel having coordinates (i, j) at the center or containing pixel (i, j), if this pixel is on or near the border. The threshold $T(i, j)$ for the pixel

with coordinate (i, j) is given by

$$T(i,j) = m(i,j) + k \cdot \sigma(i,j) \tag{1.26}$$

Sauvola and Pietaksinen's method [35] is an improvement on the Niblack method, especially for stained and badly illuminated documents. It adapts the contribution of the standard deviation. For example, in the case of text on a dirty or stained paper, the threshold is lowered. Threshold is selected as per the following criteria:

$$T(i,j) = m(i,j)\left[1 + k\left(\frac{\sigma(i,j)}{R} - 1\right)\right] \tag{1.27}$$

where R is the maximum value of the standard deviation ($R = 128$ for a grey scale document), and k is a parameter which takes positive values in the range $[0.2, 0.5]$. The local mean $m(i,j)$ and standard deviation $\sigma(i,j)$ adapt the value of the threshold according to the contrast in the local neighborhood of the pixel.

1.4.4 Noise Removal [19]

There can be different type of noises that can deteriorate quality of the image. One of the ways to remove noise is to process image in spatial domain. The term *spatial domain* refers to the aggregate of pixels composing an image[19]. Spatial domain methods are procedures that operate directly on these pixels. Spatial domain processes will be denoted by the expression

$$g(x,y) = T[f(x,y)] \tag{1.28}$$

where $f(x,y)$ is the input image, $g(x,y)$ is the processed image, and T is an operator on f. In addition,T can operate on some set of sub images of the given image, rather than on a single pixel value of the input image. The principal approach in defining a neighborhood of a point (x,y) is to use a square or rectangular subimage area centered at (x,y), as Fig. 1.9 shows. The center of the subimage is moved from pixel to pixel starting, say, at the top left corner.The operator T is applied at each location (x,y) to yield the output, g, at that location. The process utilizes only the pixels in the area of the image spanned by the neighborhood.

The general approach is to use a function of the values of f in a predefined neighborhood of (x,y) to determine the value of g at (x,y). One of the principal approaches in this formulation is based on the use of so-called *masks*

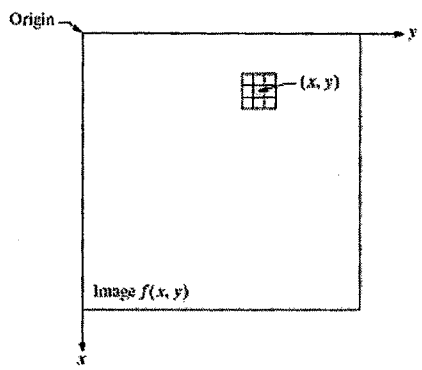


Figure 1.9: The Mechanics of Spatial Filtering

(also referred to as *filters, kernels, templates*, or *windows*). Basically, a *mask* is a small (say, 3×3) 2-D array, such as the one shown in Fig. 1.10. The size of the mask depends on the size of the neighborhood of the point (x, y) used by the operator T for computing $g(x, y)$. The values of the mask coefficients determine the nature of the process, such as image sharpening. Enhancement techniques based on this type of approach often are referred to as mask processing or filtering. Mask

The values in a filter are referred to as *coefficients*, rather than pixels. The mechanics of spatial filtering are illustrated in Fig.1.10 . The process consists simply of moving the filter mask from point to point in an image. At each point (x, y), the response of the filter at that point is calculated using a predefined relationship. For linear spatial filtering, the response is given by a sum of products of the filter coefficients and the corresponding image pixels in the area spanned by the filter mask. For the 3×3 mask shown in Fig. 1.10, the result (or response), R, of linear filtering with the filter mask at a point (x, y) in the image is

$$\begin{aligned} R = {} & w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + w(-1,1)f(x-1,y+1) \\ & + w(0,-1)f(x,y-1) + w(0,0)f(x,y) + w(0,1)f(x,y+1)) \\ & + w(1,-1)f(x+1,y-1 + w(1,0)f(x+1,y) + w(1,1)f(x+1,y+1) \end{aligned} \tag{1.29}$$

which we see is the sum of products of the mask coefficients with the corresponding pixels directly under the mask. Note in particular that the coefficient $w(0, 0)$ coincides with image value $f(x, y)$, indicating that the mask is centered at (x, y) when the computation of the sum of products takes place. For a mask of size $m \times n$, we assume that $m = 2a + 1$ and $n = 2b + 1$,where a and b are nonnegative integers.

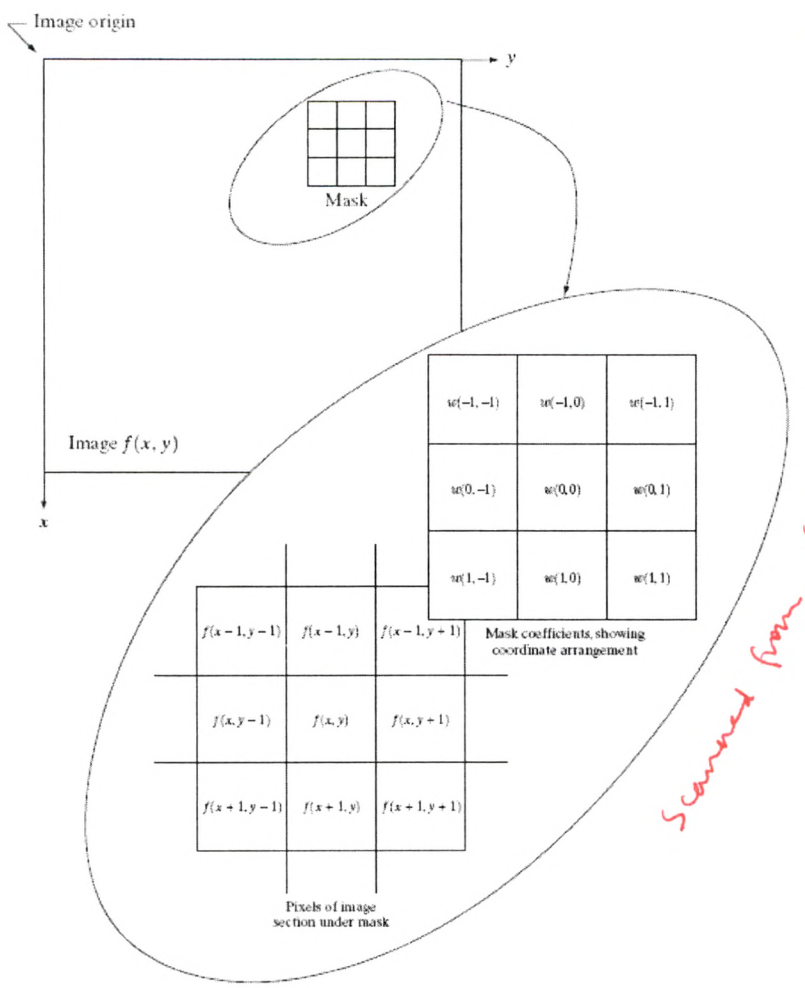


Figure 1.10: Mechanics of Linear Filter

In general, linear filtering of an image f of size $M \times N$ with a filter mask of size $m \times n$ is given by the expression:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \tag{1.30}$$

where, from the previous paragraph, $a = (m-1)/2$ and $b = (n-1)/2$. To generate a complete filtered image this equation must be applied for $x = 0, 1, 2, \ldots, M-1$ and $y = 0, 1, 2, \ldots, N-1$.

An important consideration in implementing neighborhood operations for spatial filtering is the issue of what happens when the center of the filter approaches the border of the image. Consider for simplicity a square mask of

size $n \times n$. At least one edge of such a mask will coincide with the border of the image when the center of the mask is at a distance of $(n-1)/2$ pixels away from the border of the image. If the center of the mask moves any closer to the border, one or more rows or columns of the mask will be located outside the image plane. There are several ways to handle this situation.The simplest is to limit the excursions of the center of the mask to be at a distance no less than $(n-1)/2$ pixels from the border. The resulting filtered image will be smaller than the original, but all the pixels in the filtered imaged will have been processed with the full mask. If the result is required to be the same size as the original, then the approach typically employed is to filter all pixels only with the section of the mask that is fully contained in the image. With this approach, there will be bands of pixels near the border that will have been processed with a partial filter mask. Other approaches include *padding* the image by adding rows and columns of 0's (or other constant gray level), or padding by replicating rows or columns.The padding is then stripped off at the end of the process. This keeps the size of the filtered image the same as the original, but the values of the padding will have an effect near the edges that becomes more prevalent as the size of the mask increases.The only way to obtain a perfectly filtered result is to accept a somewhat smaller filtered image by limiting the excursions of the center of the filter mask to a distance no less than $(n-1)/2$ pixels from the border of the original image.

1.4.4.1 Mean / Averaging Filter

One of the common types of noise that can corrupt an image is the impulsive noise which is reflected as sharp variation in the gray scale. One of the common ways to take care of such a noise is using *mean / averaging Filter*. The output(response) of this filter is the average of the pixel gray levels of the pixels in the neighborhood of the pixel under consideration. A 3×3 mean filter mask can be written as in Fig. 1.11

Mean Filter

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Figure 1.11: 3×3 averaging mask

This process results in an image with reduced "sharp" transitions in gray levels. Because random noise typically consists of sharp transitions in gray levels, the most obvious application of smoothing is noise reduction. However, edges (which almost always are desirable features of an image) also are characterized by sharp transitions in gray levels, so averaging filters have the undesirable

side effect that they blur edges. Another application of this type of process includes the smoothing of false contours that result from using an insufficient number of gray levels. A major use of averaging filters is in the reduction of "irrelevant" detail in an image. By "irrelevant" we mean pixel regions that are small with respect to the size of the filter mask.This latter application is illustrated later in this section.

1.4.4.2 Ordered Statistics Filter

Order-statistics filters are nonlinear spatial filters whose response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined by the ranking result.The best-known example in this category is the median filter, which, as its name implies, replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel (the original value of the pixel is included in the computation of the median).

Median filters are quite popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of impulse noise, also called *salt-and-pepper* noise because of its appearance as white and black dots superimposed on an image.

The median, ξ, of a set of values is such that half the values in the set are less than or equal to ξ, and half are greater than or equal to ξ. In order to perform median filtering at a point in an image, we first sort the values of the pixel in question and its neighbors, determine their median, and assign this value to that pixel. For example, in a 3×3 neighborhood the median is the 5th largest value, in a 5×5 neighborhood the 13th largest value, and so on.When several values in a neighborhood are the same, all equal values are grouped. For example, a 3×3 neighborhood in an image has values

10	20	20
15	20	20
20	25	100

These values in sorted order are 10, 15, 20, 20, 20, 20, 20, 25, 100. Therefore, the median is 20. Therefore, 20 will be the value for the pixel in the center(which, in this case, same as its original value) after the result of applying Median filter.

Thus, the principal function of median filters is to force points with distinct gray levels to be more like their neighbors. In fact, isolated clusters of pixels that are light or dark with respect to their neighbors, and whose area is less than $n^2/2$ (one-half the filter area), are eliminated by an $n \times n$ median filter. In this case "eliminated" means forced to the median intensity of the neighbors. Larger clusters are affected considerably less.

1.5 OCR for Indian Scripts : State of the Art

Work on the development of OCR for Indic scripts started in early 1970s. Attempts of R. M. K. Sinha *et. al.* [36] is considered to be among the first attempts to build OCR system for Devanagari script. This was followed by other researchers for example Chaudhuri and Pal [10], Kampalli *et. al.*, [6] etc.. Attempt for development of Bangla OCR system has been listed in [10]. Efforts on Telugu OCR is available in [29] and [33]. Research for many of the Indic scripts like Devanagari(used for writing Hindi, Marathi, Rajasthani, Bhojpuri, Nepali and many other languages), Gurmukhi (Punjabi), Bangla (Bengali, Assamese), Telugu (Telugu), Malayalam (Malayalam) is in an advanced stage, see for example [6], [10], [29], [33].

Research for developing Gujarati OCR started much later than the research level attempts of developing OCR systems for other Indian scripts like Bangla and Devanagari started. Due to common roots of Indo-Aryan scripts and similarity due to that, the work that have been done for other script for various subtasks like line/word segmentation [6], [31], recognition [10], [6], [29] has been useful in this research work. Some ideas from work in English handprinted character recognition [8] are also used for Gujarati document image analysis.

Our first exposure to Gujarati character recognition was from the work published in ICDAR 1999 [1], where an experiment to recognize 10 pre-segmented Gujarati characters using Hu invariant moments as feature extractors, and K-Nearest Neighbor classification was presented. The accuracy reported was very low and there was no systemic approach to the problem of Gujarati Document Analysis or Recognition. A recent Ph.D. thesis [47] shows use of mathematical concepts like wavelets and artificial neural networks for Gujarati character recognition.

1.6 Gujarati Script from OCR Perspective

Gujarati script is used to write the Gujarati language spoken by about 50 million people (also called Gujaraties) in the western part of India. Apart from the native speakers of the state of Gujarat in India, Gujarati speaking diaspora is spread across all parts of India and in many parts of the world. Gujarati literature is not only of interest to the Gujaratis but also to the researchers abroad [17], [18], [16].

Gujarati has 12 vowels and 34+2* consonants. Apart from these basic symbols, other symbols called "vowel modifiers" are used to denote the attachment of vowels with the core consonants. Consonant-Vowel combinations occur very often in most of the Indic languages including Gujarati. This is denoted by attaching a symbol, unique for each vowel, to the consonant, called a Dependent Vowel Modifier or *Matra*. The *matra* can appear before, after, above or below the core consonant.

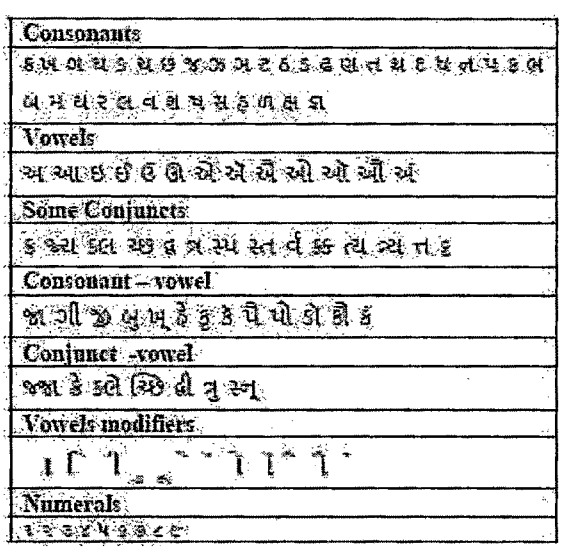


Figure 1.12: Gujarati Script

In addition to basic consonants, like most Indic scripts, Gujarati also uses consonant clusters (conjuncts). That is, consonants without the inherent vowel sound are combined and thereby leading to three possibilities for the shape of the resulting conjuncts:

1. Conjunct shape is derived by connecting a part of the preceding consonant to the following one

*conjuncts /ksha/ and /jya/ are treated as if they are basic consonants in Gujarati script

ક + ્ + ત = ક્ત , ટ + ્+ ટ = ટ્ટ

2. The conjunct can take completely different shape

ઙ+ ્ + ચ =ઙ્ચ , ક+ ્ +ષ = ક્ષ , ઙ+ ્ + વ =ઙ્વ

3. Addition of some mark indicating conjunct formation in upper/middle/lower zone (mainly conjuncts involving /r/).

ક + ્ + ર= ક , ર + ્ + વ = ર્વ , ટ + ્ +ર = ટ્ર

Moreover, conjuncts may themselves occur in half forms. [22] gives detailed description on Gujarati script with the rules to form conjuncts and other modifications that might take place in the shapes of basic consonant symbols.

It can be seen that the shapes of many Gujarati characters are similar to those of the phonetically corresponding characters of Devanagari script. As in the case of other Indic scripts, Gujarati also does not have the distinction of Lower and Upper Cases. Inspite of these similarities with the Devanagari script, Gujarati script has many distinct characteristics and the most important of them is **absence of so-called *shirorekha(header line)*** in the script and differences in the shapes of many of the consonants and vowels, etc.

Similar to the text written in Devanagari or Bangla script, text in Gujarati script can also be divided into three logical zones: Upper, Middle and Lower as shown in Fig.1.13.

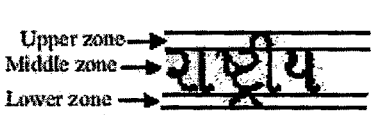


Figure 1.13: Logical Zones

1.7 Motivation

Gujarati is one of the official languages of India which belongs to the group of Indo-Aryan languages and is written in Gujarati script. The language has literary tradition going back to ten centuries [40] and is also privileged by the original writings of Mahatma Gandhi [17],[18] and others. Another relevant fact is that the oldest running published newspaper in India is a Gujarati daily, "Mumbai Samachar", published since 1822. However, it is a surprising

fact that, very little of this wealth of Gujarati literature is available in an electronic form which can allow searching and indexing. For recreating this wealth of existing literary work in machine readable form, it is impractical to think of getting it typed again as it is time consuming and prone to errors. The solution to this problem lies in building robust Optical Character Recognition(OCR) Systems for Gujarati script. However, complexity of Gujarati script poses several challenges in building such a system.

It is clear from the above discussion that mathematics is applied at various stages of OCR. It may also be noted that Gujarati OCR was almost an untouched area at the time we started this work. There was only one publication in 1999 discussing use of mathematical techniques for Gujarati character recognition [1]. Some work on Gujarati OCR was reported during the course of this research by other researchers [38],[46],[47], but all the published work was proposing some mathematical techniques to handle recognition subtasks of the entire OCR system. It is also important to add that all of these was considering only a small subset of the entire set of glyphs that needs to be recognized in order to build the complete system. Further, there were no attempts reported to address the script specific aspect of segmentation and document image analysis.

These facts motivated us to explore the applications of various mathematical techniques to attack this problem in totality. Various script specific tasks and the application of various mathematical techniques are described in the subsequent chapters.

It is important to note at this point that as a part of this research work we have applied mathematical techniques to solve some of the untouched problems which are specific to Gujarati. For the tasks which are script independent we have tested some of the available techniques used by other scripts and selected ones appropriately.

1.8 Organization of Thesis

After this detailed introduction listing various mathematical techniques applied for carrying out some of the subtasks of an OCR system, subsequent part of the thesis is logically divided into two parts. First part consisting of chapter 2, chapter 3 and chapter 4 focuses on applications of mathematical techniques for segmentation and document analysis and the second one is devoted to discuss techniques used to Gujarati character recognition and text

generation.

Chapter 2 describes various techniques for document segmentation ranging from block level segmentation to connected component extraction. These techniques are more or less script independent. Hence we have adopted some of the existing techniques developed for other scripts with appropriate modifications for Gujarati.

Chapter 3 is about one of the two techniques for zone boundary identification, developed as a part of this research. In this chapter we elaborate on the need for a robust zone separation algorithm for Gujarati script and propose a solution for the same based on the simple concept of slope of a line.

Chapter 4 describes another approach to address the problem of zone boundary identification. In this chapter we introduce a concept of vertical touching and model the problem of zone separation as touching character segmentation problem and propose a solution based on dynamic programming.

Chapter 5 presents feature extraction methods that are tested during the course of research. A detailed study of these methods with necessary analysis of their representation capabilities are discussed in this chapter.

Chapter 6 is devoted to discussion of nearest neighbor and General Regression Neural Network (GRNN) as classifiers along with their performance for Gujarati character classification.

In Chapter 7, we propose a finite state machine based approach to generate text from the recognized components.

Chapter 8, the last chapter of the thesis, is devoted to present results of various experiments carried out by varying parameters for each and every subtask. This chapter concludes the thesis by presenting a work flow for integrating these subsystems in to a software package.

Appendix following chapter 8 consists of list of publications which resulted out of this research, Unicode chart for Gujarati script and list of references used during the course of this research.