

## **ABSTRACT**

Text detection and extraction is very useful for keyword-based image search, automatic video logging, and automatic text-based image indexing. It is also useful for the tourists to know the text written on name plates and sign boards, generally which is in local language. It's also useful to detect the text in news streaming to know the news updates. Text embedded in the image or video contains very useful information like the name of the person, title, location and sometimes brief description of the image. However, variations of text due to differences in size, style, orientation, and alignment, as well as low image contrast and complex background make the problem of automatic text extraction extremely challenging.

In this project, our main intention is to implement system that will detect the multi lingual text from images and the primary emphasis is on images containing Telugu, Hindi and English text. We also evaluated the performance of this system with that of edge based and connected component based methods.

## TABLE OF CONTENTS

Certificate	i
Declaration	ii
Acknowledgement	iii
Abstract	vii
List of Figures	xi
List of Tables	xii
1. INTRODUCTION	1
1.1. Motivation	1
1.2. Problem Specification	2
1.3. Methodologies	3
2. LITERATURE SURVEY	4
2.1. Fundamentals In Image Processing	4
2.2. Principle of Text	6
2.3. Text Information Extraction	6
2.4. Text in Images	8
2.5. Text Extraction Methods	11
2.6. Related Work	12
3. DESIGN	15
3.1. Background removal in the DCT domain	15
3.2. Feature extraction	15
3.3. Classification	15
3.4. Merging of Text Blocks	16
3.5. Post processing	16
3.6. Design Diagrams	17
4. REQUIREMENTS SPECIFICATIONS	20
5. IMPLEMENTATION	21
5.1. Back removal/suppression in the DCT domain	21
5.2. Feature Extraction	22
5.3. Classification	23
5.4. Merging of Text Blocks to Detect Text Regions	24
5.5. Refinement of Text Regions	25
6. RESULTS AND OBSERVATIONS	28
6.1. Results	28
6.2. Observations	35

7. CONCLUSIONS AND FUTURE SCOPE	40
7.1 Conclusions	40
7.2 Future Scope	40
• REFERENCES	42
• APPENDIX	43
○ Matlab	43
○ Test Images	46

## LIST OF FIGURES

Figure 2.1 Representation of an Image	5
Figure 2.2 Architecture of Text Information Extraction	6
Figure 2.3 Gray-scale document images	9
Figure 2.4 Multi-color document images	9
Figure 2.5 Images with caption text	9
Figure 2.6 Scene text images	10
Figure 3.1 Block Diagram	15
Figure 3.2 Data Flow Diagram	17
Figure 3.3 Class Diagram	18
Figure 3.4 Sequence Diagram	19
Figure 5.1 High Pass Filter for Background Removal using DCT	22
Figure 6.1 The process of text extraction	28
Figure 6.2 50x50 blocks of preprocessed image in figure 6.1b	30
Figure 6.5 Text Extraction results of processing of low resolution natural scene images dealing with various issues	33
Figure 6.6 Overall results of proposed model for text localization	35
Figure 6.7 Result Images of Three detection Methods	37
Figure 6.8 Performance Measure of three Methods	39

## LIST OF TABLES

Table 6.1 Preprocessing of first 8x8 block of an image in figure 6.1 a	29
Table 6.2 Feature Matrix $D$ containing extracted features of image of size 240x320 given in figure 6.1	30
Table 6.3 Vector $B$ showing coordinate values of identified text blocks	31
Table 6.5 The Performance of the system of processing different images given in figure 6.5dealing with various issues	34
Table 6.7 Precision and Recall rates of the overall system	36
Table 6.8 Comparison of the three methods on the common test set	38

# 1. INTRODUCTION

Recent studies in the field of computer vision and pattern recognition showed a great amount of interest in content retrieval from images and videos. This content can be in the form of objects, color, texture, shape as well as the relationships between them. The semantic information provided by an image can be useful for content based image retrieval, as well as for indexing and classification purposes [4, 10]. As stated by Jung, Kim and Jain in [4], text data is particularly interesting, because text can be used to easily and clearly describe the contents of an image. Since the text data can be embedded in an image or video in different font styles, sizes, orientations, colors, and against a complex background, the problem of extracting the candidate text region becomes a challenging one [4]. Also, current Optical Character Recognition (OCR) techniques can only handle text against a plain monochrome background and cannot extract text from a complex or textured background [7].

## 1.1 Motivation

As the people move across world for business, field works and/or pleasure, they find it difficult to understand the text written on display boards in foreign environment. In such a scenario, people either look for guides or intelligent devices that can help them in providing translated information to their native language. As most of the individuals carry camera embedded, hand held devices such as mobile phones and PDA's, there is a possibility to integrate technological solutions into such systems in order to provide facilities for automatically understanding display boards in foreign environment. These facilities may be provided as an integral solution through web service as necessary computing function, which are not available in hand held systems. Such web based hand held systems must be enabled to capture natural scene images containing display boards and query the web service to retrieve translated localized information of the text written on display boards.

The written matter on display/name boards provides information necessary for the needs and safety of people, and may be written in languages unknown. And the written matter can be street names, restaurant names, building names, company names, traffic directions, warning signs etc. Hence, lot of commercial and academic interest is veered towards development of techniques for web service based hand held

systems useful in understanding written text in display boards. There is a spurt of activity in development of web based intelligent hand held tour guide systems, blind assistants to read written text and Location aware computing systems and many more in recent years. A few such works are presented in the following and a more elaborate survey of related works is given in the next section. *A point by photograph paradigm* where users can specify an object by simply taking picture to retrieve matching images from the web is found in [5]. The *comMotion* is a location aware hand held system that links personal information to locations. It reminds users about shopping list when he/she nears a shopping mall [6]. At Hewlett Packard (HP), mobile Optical Character Reading (OCR) applications were developed to retrieve information related to the text image captured through a pen-size camera [8]. Mobile phone image matching and retrieval has been used by insurance and trading firms for remote item appraisal and verification with a central database [4].

The image matching and retrieval applications cannot be embedded in hand held devices such as mobile phones due to limited availability of computing resources, hence such services are being developed as web services. The researchers have also worked towards development of web based intelligent hand held tour guide systems. The cyberGuide [5] is an intelligent hand held tour guide system, which provides the information based on user's location. The cyberGuide continuously monitors the users location using Global Positioning System (GPS) and provides new information at the right time. Museums could provide these tour guides to visitors allowing them to take personalized tours observing any displayed object. As the visitors move across museum floors, the information about the location is pushed to hand held tour guides. The research prototypes used to search information about an object image captured by cameras embedded in mobile phones are described in [6, 7]

## **1.2 Problem Specification**

The purpose of this project is to implement a texture based text detection approach proposed in [1] which aims at effective detection and localization of text from the natural images focusing on images with kannada text. We intend to apply this method to the images with telugu ,hindi and English text. The major goal of our project is extend this method for text localization from the images of other languages with little modifications. In this project we will implement the Texture Based Text detection method [1] with little

modifications and then present the results of performance evaluation on comparison with the edge based detection method [2,3] and connected component based method[4].

### **1.3 Methodologies**

The state of art hand held systems available across the world are not automated for understanding written text on display boards in foreign environment. Scope exists for exploring such possibilities through automation of hand held systems. One of the very important processing steps for development of such systems is automatic detection and extraction of text regions from low resolution natural scene images prior to further analysis. The written text provides important information and it is not an easy problem to reliably detect and localize text embedded in natural scene images [8]. The size of the characters can vary from very small to very big. The font of the text can be different. Text present in the image may have multiple colors. The text may appear in different orientation. Text can occur in a complex background. And also the textual and other information captured is affected by significant degradations such as perspective distortion, blur, shadow and uneven lighting. Hence, the automatic detection and segmentation of text is a difficult and challenging problem. Reported works have identified a number of approaches for text localization from natural scene images. The existing approaches are categorized as connected component based, edge based and texture based methods. Connected component based methods use bottom up approach to group smaller components into larger components until all regions are identified in the image. A geometrical analysis is later needed to identify text components and group them to localize text regions. Edge based methods focus on the high contrast between the background and text and the edges of the text boundary are identified and merged. Later several heuristics are required to filter out non-text regions. But, the presence of noise, complex background, and significant degradation in the low resolution natural scene image can affect the extraction of connected components and identification of boundary lines, thus making both the approaches inefficient. Texture analysis techniques are good choice for solving such a problem as they give global measure of properties of a region.



## 2. LITERATURE SURVEY

### 2.1 FUNDAMENTALS IN IMAGE PROCESSING

#### 2.1.1 Image Representation

An image (from Latin *imago*) is an artifact, for example a two-dimensional picture that has a similar appearance to some subject— usually a physical object or a person. An image may be defined as a two-dimensional function  $f(x, y)$ , where  $x$  and  $y$  are spatial coordinates, and the amplitude of ‘ $f$ ’ at any pair of coordinates  $(x, y)$  is called gray level or intensity of the image at that point. Images may be acquired from many sources, including a disk file, the network, a CD, and so on. Images may be acquired, processed, and immediately displayed, or written to a disk file for display at a later time.

Image data is, conceptually, a three-dimensional array of pixels, as shown in figure. Each of the three arrays in the example is called a *band*. The number of rows specifies the image height of a band, and the number of columns specifies the image width of a band.

Monochrome images, such as a gray scale image, have only one band. Color images have three or more bands, although a band does not necessarily have to represent color. For example, satellite images of the earth may be acquired in several different spectral bands, such as red, green, blue, and infrared.

In a color image, each band stores the red, green, and blue (RGB) components of an additive image, or the cyan, magenta, and yellow (CMY) components of a three-color subtractive image, or the cyan, magenta, yellow, and black (CMYK) components of a four-color subtractive image. Each pixel of an image is composed of a set of *samples*. For an RGB pixel, there are three samples; one each for red, green, and blue.

An image is sampled into a rectangular array of pixels. Each pixel has an  $(x, y)$  coordinate that corresponds to its location within the image. The  $x$  coordinate is the pixel's horizontal location; the  $y$  coordinate is the pixel's vertical location.

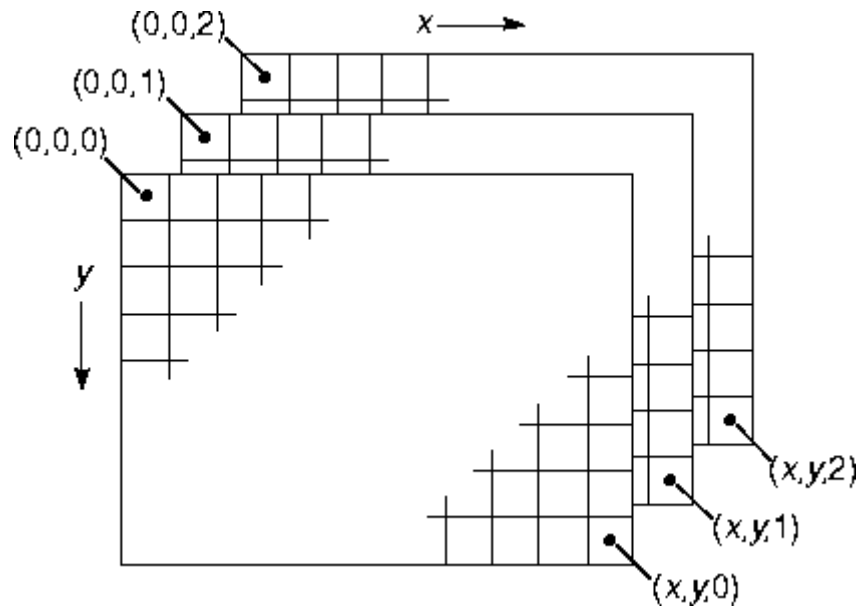


Figure. 2.1 Representation of an Image

### 2.1.2 Image File Formats

These are standardized means of organizing and storing images. Image files are composed of either pixel or vector (geometric) data that are rasterized to pixels when displayed (with few exceptions) in a vector graphic display. The pixels that compose an image are ordered as a grid (columns and rows) each pixel consists of numbers representing magnitudes of brightness and color.

Some of the image formats are:

- JPEG (Joint Photographic Experts Group)
- PNG (Portable Network Graphics)
- GIF (Graphics Interchange Format)
- BMP file format (Windows bitmap)
- TIFF (Tagged Image File Format)

### 2.1.3 Digital Image Processing

Digital image processing is the use of computer algorithms to perform image processing on digital images. As a subcategory or field of digital signal processing, digital image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since

images are defined over two dimensions (perhaps more) digital image processing may be modeled in the form of Multidimensional Systems [11].

### 2.1.3.1 Applications

One of the applications of image processing is to categorize images according to their source like Gamma-Ray imaging, Imaging in the ultra-violet band, Imaging in the infrared and visible bands, Imaging in the microwave band, Imaging in the radio band. Medical imaging, videophone, character recognition are some more.

## 2.2 Principle of Text

Text extraction is the ability to extract (pull out) text from a document or image. A TIE system receives an input in the form of a still image or a sequence of images. The images can be in gray scale or color and the text in the images. The images can be in gray scale or color, compressed or uncompressed, and the text in the images may or may not move.

## 2.3 Text Information Extraction (Tie)

The TIE problem can be divided into the following sub-problems: (i) Detection, (ii) Localization, (iii) Tracking, (iv) Extraction and Enhancement, and (v) Recognition (OCR)

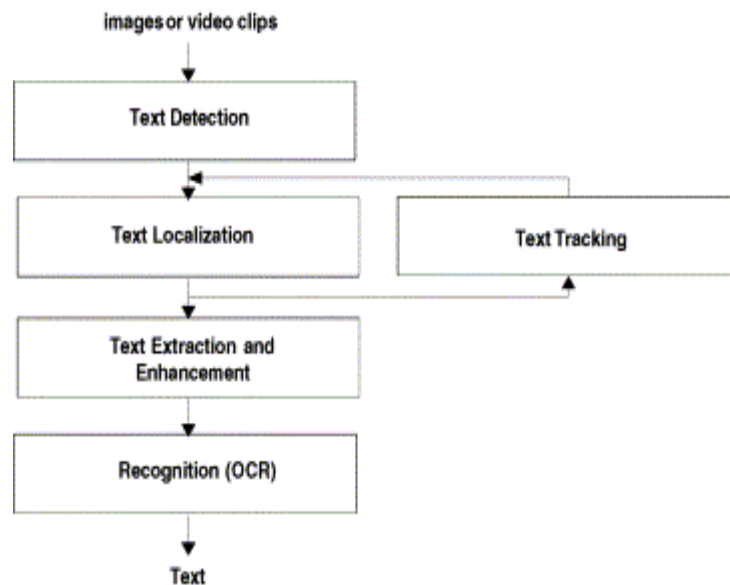


Figure 2.2 Architecture of Text Information Extraction

*Text detection* refers to the determination of the presence of text in a given frame (normally text detection is used for a sequence of images).

*Text localization* is the process of determining the location of text in the image and generating bounding boxes around the text.

*Text tracking* is performed to reduce the processing time for text localization and to maintain the integrity of position across adjacent frames. Although the precise location of text in an image can be indicated by bounding boxes, the text still needs to be segmented from the background to facilitate its recognition. This means that the extracted text image has to be converted to a binary image and enhanced before it is fed into an OCR engine.

*Text extraction* is the stage where the text components are segmented from the background.

*Text Enhancement* of the extracted text components is required because the text region usually has low resolution and is prone to noise. Thereafter, the extracted text images can be transformed into plain text using OCR technology.

### **2.3.1 Applications**

Text extraction from images finds many useful applications in document analysis, vehicle license plate extraction, content based image retrieval, text- based image indexing, video content analysis, industrial automation etc and many applications have become realities in recent years [8]. Educational and training video and TV programs such as news contain mixed text-picture graphics regions. Region classification is helpful in object-based compression, manipulation and accessibility. Also, text regions may carry useful information about the visual content.

However, as mentioned earlier, due to the variety of fonts, sizes, styles, orientations, alignment effects of uncontrolled illuminations, reflections, shadows, the distortion due to perspective projection as well as the complexity of image background, automatic localizing and extracting text is a challenging problem.

## 2.4 Text in Images

A variety of approaches to text information extraction (TIE) from images and video have been proposed for special applications including page segmentation [12, 14], address block location [13], license plate location [8, 9], and content-based image/video indexing [5, 12]. In spite of such extensive studies, it is still not easy to design a general-purpose TIE system. This is because there are so many possible sources of variation when extracting text from a shaded or textured background, from low-contrast or complex images, or from images having variations in font size, style, color, orientation, and alignment. These variations make the problem of automatic TIE extremely difficult. Figures 3–6 show some examples of text in images. Page layout analysis usually deals with document images 1 (Figure. 1). Readers may refer to papers on document segmentation/analysis [12, 14] for more examples of document images. Although images acquired by scanning book covers, CD covers, or other multi-colored documents have similar characteristics as the document images (Figure. 2), they cannot be directly dealt with using a conventional document image analysis technique. Accordingly, this survey distinguishes this category of images as multi-color document images from other document images. Text in video images can be further classified into caption text (Figure. 3), which is artificially overlaid on the image, or scene text (Figure. 4), which exists naturally in the image. Some researchers like to use the term ‘graphics text’ for scene text, and ‘superimposed text’ or ‘artificial text’ for caption text [13]. It is well known that scene text is more difficult to detect and very little work has been done in this area. In contrast to caption text, scene text can have any orientation and may be distorted by the perspective projection. Moreover, it is often affected by variations in scene and camera parameters such as illumination, focus, motion, etc.

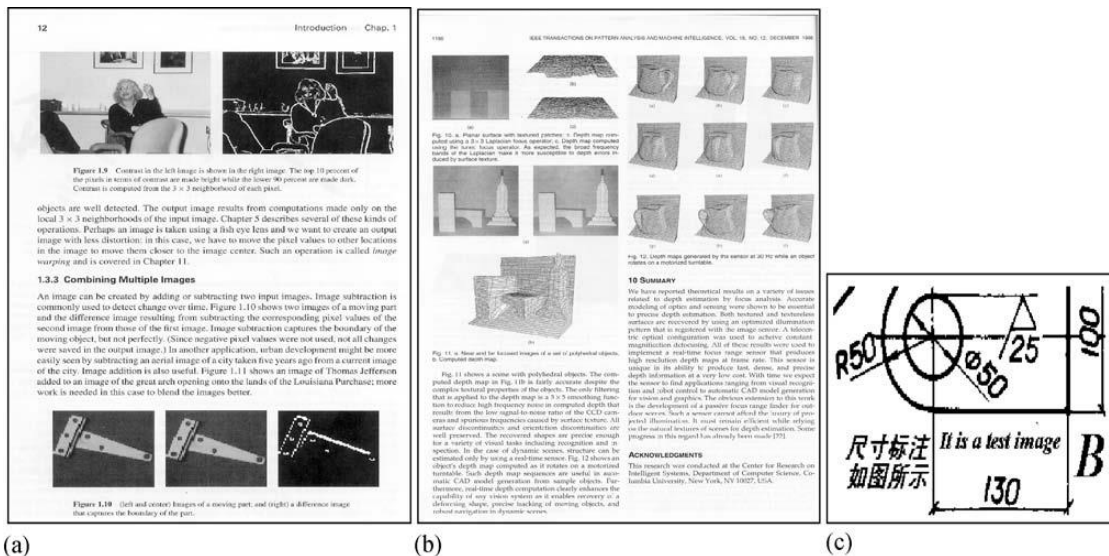


Figure.2.3: Gray-scale document images: (a) single-column text from a book, (b) two-column page from a journal (IEEE Transactions on PAMI), and (c) an electrical drawing (courtesy Lu [14]).

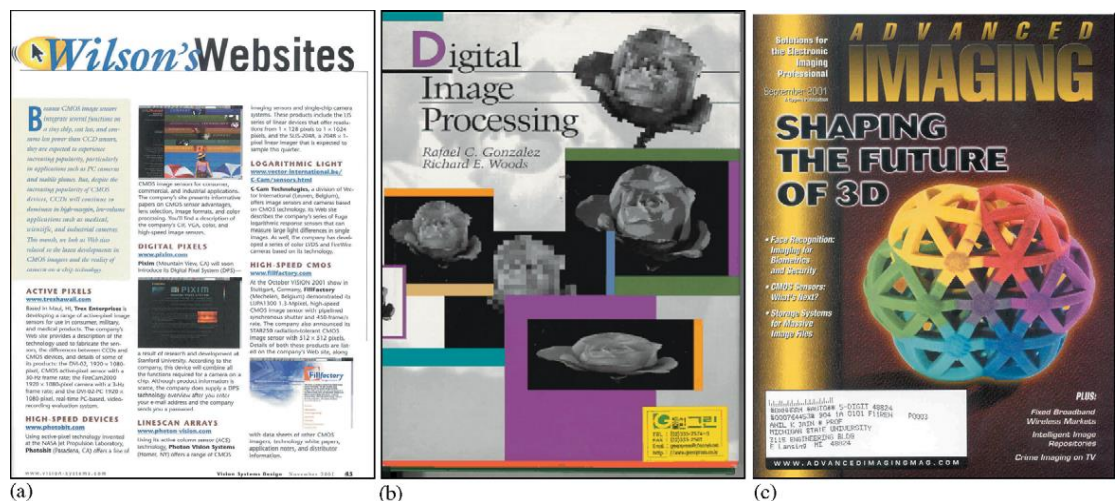


Figure.2.4. Multi-color document images: each text line may or may not be of the same color



Figure.2.5. Images with caption text: (a) shows captions overlaid directly on the background. (b) and (c) text in frames for better contrast, (c) text string that is polychrome.





Figure 2.6. Scene text images: Images with variations in skew, perspective, blur, illumination, and alignment.

Before we attempt to classify the various TIE techniques, it is important to define the commonly used terms and summarize the characteristics of text. Table 1 shows a list of properties that have been utilized in recently published algorithms. Text in images can exhibit many variations with respect to the following properties:

#### 1. Geometry:

- *Size*: Although the text size can vary a lot, assumptions can be made depending on the application domain.
- *Alignment*: The caption texts appear in clusters and usually lie horizontally, although sometimes they can appear as non-planar texts as a result of special effects. This does not apply to scene text, which has various perspective distortions. Scene text can All these properties may not be important to every application can be aligned in any direction and can have geometric distortions (Figure 2. 4).
- *Inter-character distance*: characters in a text line have a uniform distance between them.

#### 2. Color:

The characters tend to have the same or similar colors. This property makes it possible to use a connected component-based approach for text detection. Most of the research reported till date has concentrated on finding ‘text strings of a single color (monochrome)’. However, video images and other complex color documents can contain ‘text strings with more than two colors (polychrome)’ for effective visualization, i.e., different colors within one word.

### 3. Motion:

The same characters usually exist in consecutive frames in a video with or without movement. This property is used in text tracking and enhancement. Caption text usually moves in a uniform way: horizontally or vertically. Scene text can have arbitrary motion due to camera or object movement.

4. Edge: Most caption and scene text is designed to be easily read, thereby resulting in strong edges at the boundaries of text and background.

5. Compression: Many digital images are recorded, transferred, and processed in a compressed format. This improves the speed of TIE system.

## **2.5 Text Extraction Methods**

Text extraction in images includes five stages, among which text detection and text localization are closely related and more challenging stages which attract the attention of most researchers. The goal of the two stages is to generate accurate bounding boxes of all text objects in images and video frames and provide a unique identity to each text. In this section, the recent techniques focused on text detection and localization is discussed.

### **2.5.1 Edge based detection method**

Edge based method focus on high contrast between the background and text and the edges of the text boundary are identified and merged. Later several heuristics are required to filter out the non - text regions.

### **2.5.2 Connected-component based detection method**

Connected component based methods use bottom up approach to group smaller components into larger components until all regions are identified in the image. A geometrical analysis is later needed to identify text components and group them to localize text regions.

### **2.5.3 Texture based method**

Texture-based methods use the observation that the text in the images has distinct textural properties that distinguish them from the background. The techniques based on Gabor filters, Wavelet, FFT, spatial variance, etc. can be used to detect the textural properties of a text region in an image.



#### **2.5.4 Morphological Based Method**

Mathematical morphology is a topological and geometrical based approach for image analysis. It provides powerful tools for extracting geometrical structures and representing shapes in many applications. Morphological feature extraction techniques have been efficiently applied to character recognition and document analysis. It is used to extract important text contrast features from the processed images. The feature is invariant against various geometrical image changes like translation, rotation, and scaling. Even after the lighting condition or text color is changed, the feature still can be maintained. This method works robustly under different image alterations.

#### **2.6 Related work**

Various methods have been proposed in the past for detection and localization of text in images and videos. These approaches take into consideration different properties related to text in an image such as color, intensity, connected-components, edges etc. These properties are used to distinguish text regions from their background and/or other regions within the image. The algorithm proposed by Wang and Kangas in [5] is based on color clustering. The input image is first pre-processed to remove any noise if present. Then the image is grouped into different color layers and a gray component. This approach utilizes the fact that usually the color data in text characters is different from the color data in the background. The potential text regions are localized using connected component based heuristics from these layers. Also an aligning and merging analysis (AMA) method is used in which each row and column value is analyzed [5]. The experiments conducted show that the algorithm is robust in locating mostly Chinese and English characters in images. Some false alarms occurred due to uneven lighting or reflection conditions in the test images.

The text detection algorithm in [6] is also based on color continuity. In addition it also Uses multi-resolution wavelet transforms and combines low as well as high level image features for text region extraction. The text finder algorithm proposed in [7] is based on the frequency, orientation and spacing of text within an image. Texture based segmentation is used to distinguish text from its background. Further a bottom-up ‘chip generation’ process is carried out which uses the spatial cohesion property of

text characters. The chips are collections of pixels in the image consisting of potential text strokes and edges. The results show that the algorithm is robust in most cases, except for very small text characters that are not properly detected. Also in the case of low contrast in the image, misclassifications occur in the texture segmentation.

A focus of attention based system for text region localization has been proposed by Liu and Samarabandu in [8]. The intensity profiles and spatial variance is used to detect text regions in images. A Gaussian pyramid is created with the original image at different resolutions or scales. The text regions are detected in the highest resolution image and then in each successive lower resolution image in the pyramid. The approach used in [9, 11] utilizes a support vector machine (SVM) classifier to segment text from non-text in an image or video frame. Initially text is detected in multi scale images using edge based techniques, morphological operations and projection profiles of the image [11]. These detected text regions are then verified using wavelet features and SVM. The algorithm is robust with respect to variance in color and size of font as well as language.

A few state of the art approaches that use texture features for text localization have been summarized here; the use of horizontal window of size  $1 \times 21$  (Mask size) to compute the spatial variance for identification of edges in an image, which are further used to locate the boundaries of a text line is proposed in [9]. However, the approach will only detect horizontal components with a large variation compared to the background and a processing time of 6.6 seconds with  $256 \times 256$  images on SPARC station 20 is reported. The Vehicle license plate localization method that uses similar criteria is presented in [10]. It uses time delay neural networks (TDNNs) as a texture discriminator in the HSI color space to decide whether the window of an image contains a license plate number. The detected windows are later merged for extracting license plates. A multi-scale texture segmentation schemes are presented in [11-12]. The methods detect potential text regions based on nine second-order Gaussian derivatives and is evaluated for different images including video frames, newspapers, magazines, envelopes etc. The approach is insensitive to the image resolution and tends to miss very small text and gives a localization rate of 90%.

A methodology that uses frequency features such as the number of edge pixels in horizontal and vertical directions and Fourier spectrum to detect text regions in real scene images is discussed in [13]. The texture-based text localization method using

Wavelet transform is proposed in [14]. The techniques for text extraction in complex color images, where a neural network is employed to train a set of texture discrimination masks that minimize the classification error for the two texture classes: text regions and non-text regions are reported in [15].

Learning-based methods for localizing text in documents and video are proposed in [15] and [13]. The method [18] is evaluated for various video images and the text localization procedure required about 1 second for processing a 352x240 image on Sun workstation. And for text detection a precision rate of 91% and a recall rate of 92.8% is reported. This method was subsequently enhanced for skewed text in [9]. A work similar to the proposed method which uses DCT coefficients to capture textural properties for caption localization is presented in [8]. The authors claim that the method is very fast and gives a detection rate of 99.17%. However, the precise localization results are not reported. In recent years, several approaches for sign detection, text detection and segmentation from natural images are also reported [14].

Out of many works cited in the literature it is generally agreed that the robustness of texture based methods depends on texture features extracted from the window/block or the region of interest that are used by the discriminant functions for classification decisions. The probability of misclassification is directly related to the number and quality of details available in texture features. Hence, the extracted texture features must give sufficient information to distinguish text from the background. And also suppression/removal of background information is an essential preprocessing step needed before extracting distinguishable texture features to reduce the probability of misclassification. But most of the works cited in the literature directly operate on the image without suppressing the background. Hence, there is a scope to explore such possibilities. The present method which we choose to implement performs preprocessing on the image for suppressing the uniform background in the DCT domain and further uses texture features for text localization. The detailed description of the proposed methodology is given in the next section.

### 3. DESIGN

This is a texture based method and can operate on low resolution images. The method uses high pass filter in the DCT domain to suppress the background and homogeneity, contrast are the texture features. For suppressing the background the image is divided into 8x8 blocks whereas for computing the texture features, the image has been divided into 50x50 blocks. The method consists of five phases. Architecture view of the five phases in this texture based method is shown in the figure 3.1

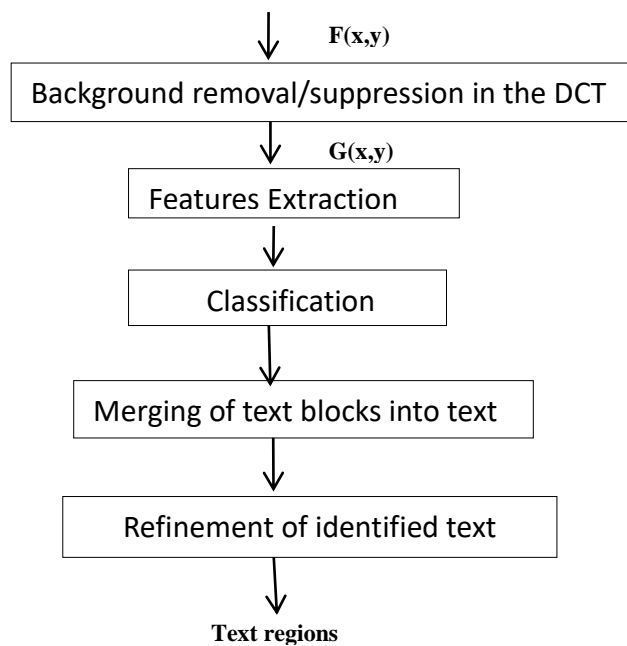


Figure3.1 Block Diagram

#### 3.1 Background removal in the DCT domain:

The removal of background from the image resulting from the sources such as roads, trees, walls, etc. The method uses DCT coefficients to remove the background. The input image is divided into 8x8 blocks; find the DCT coefficient of each block. In the resultant coefficients, the values from top to bottom indicates horizontal variations and the values from left to right indicate the vertical variations. The top left coefficient corresponds to DC component, represents low frequency component and contains most of the energy. Hence, the background can be removed by applying a

high pass filter that ignores the DC component. And then by using the new coefficients we'll construct the image by applying inverse DCT.

### 3.2 Feature Extraction:


In this phase, the texture features such as homogeneity and contrast are computed for each 50x50 block. We'll compute these features in four orientations viz.  $0^0$ ,  $45^0$ ,  $90^0$  and  $135^0$  degrees.

*Homogeneity* gives information about how uniform the block is.

*Contrast* gives information about how much the block can be focused.

### 3.3 Classification:

From the above features, the text components can be classified by using following rules:

 Text block, if for all  $h_i \leq t_1$  and  $c_i \leq t_2$   
No-text block, otherwise

The values for  $t_1$  and  $t_2$  are set empirically. These values are highly depending on the detection process.

### 3.4 Merging of Text Blocks:

This step is to perform the Localization stage of Text information Extraction system where obtained text blocks in the previous phase are merged to form the text regions. In this process among detected blocks, which are row/column wise connected are merged And correspondingly, the coordinates of the text blocks are updated.

### 3.5 Post Processing:

In this phase, any unprocessed block has been processed. For example, if the resolution of the image is 230x220 then the unprocessed 30 rows and 20 columns processed, compute the average contrast and by using some heuristics we can classify them. In this phase, we also refine the text blocks by combining entire size or some rows/cols of the undetected adjacent block which contain portions of the missed text .

### 3.6 DESIGN DIAGRAMS

#### 3.6.1 Data Flow Diagram

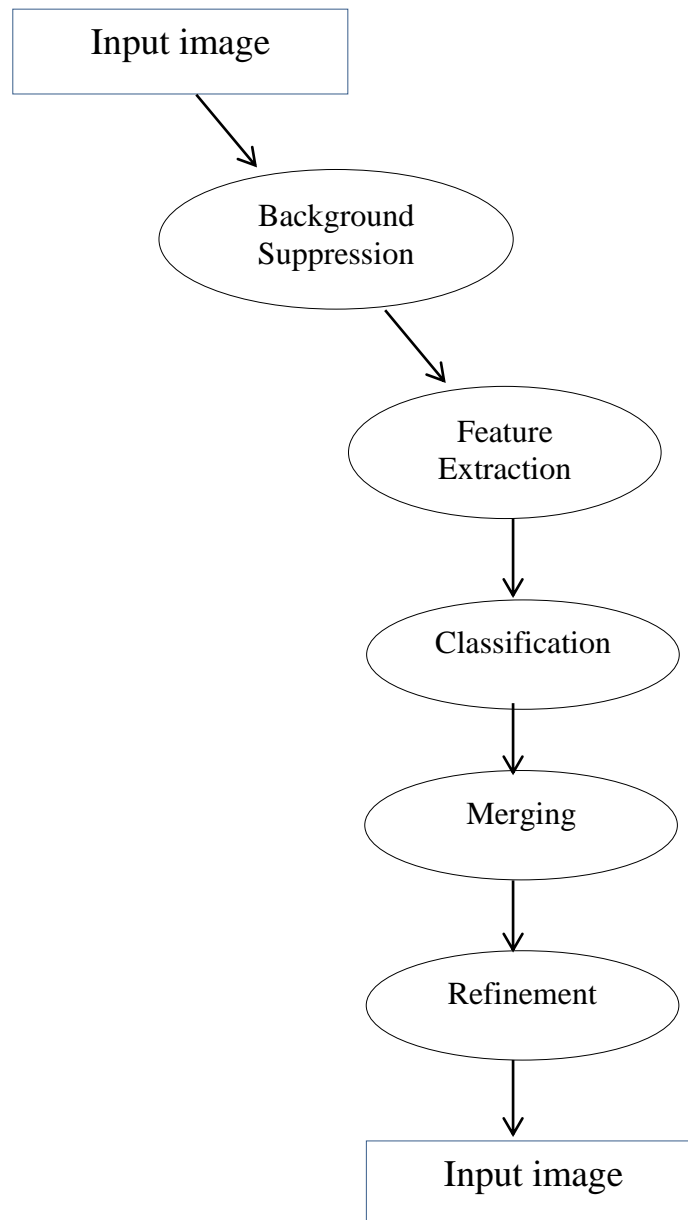


Figure3.2 Data Flow Diagram

### 3.6.2 Class Diagram

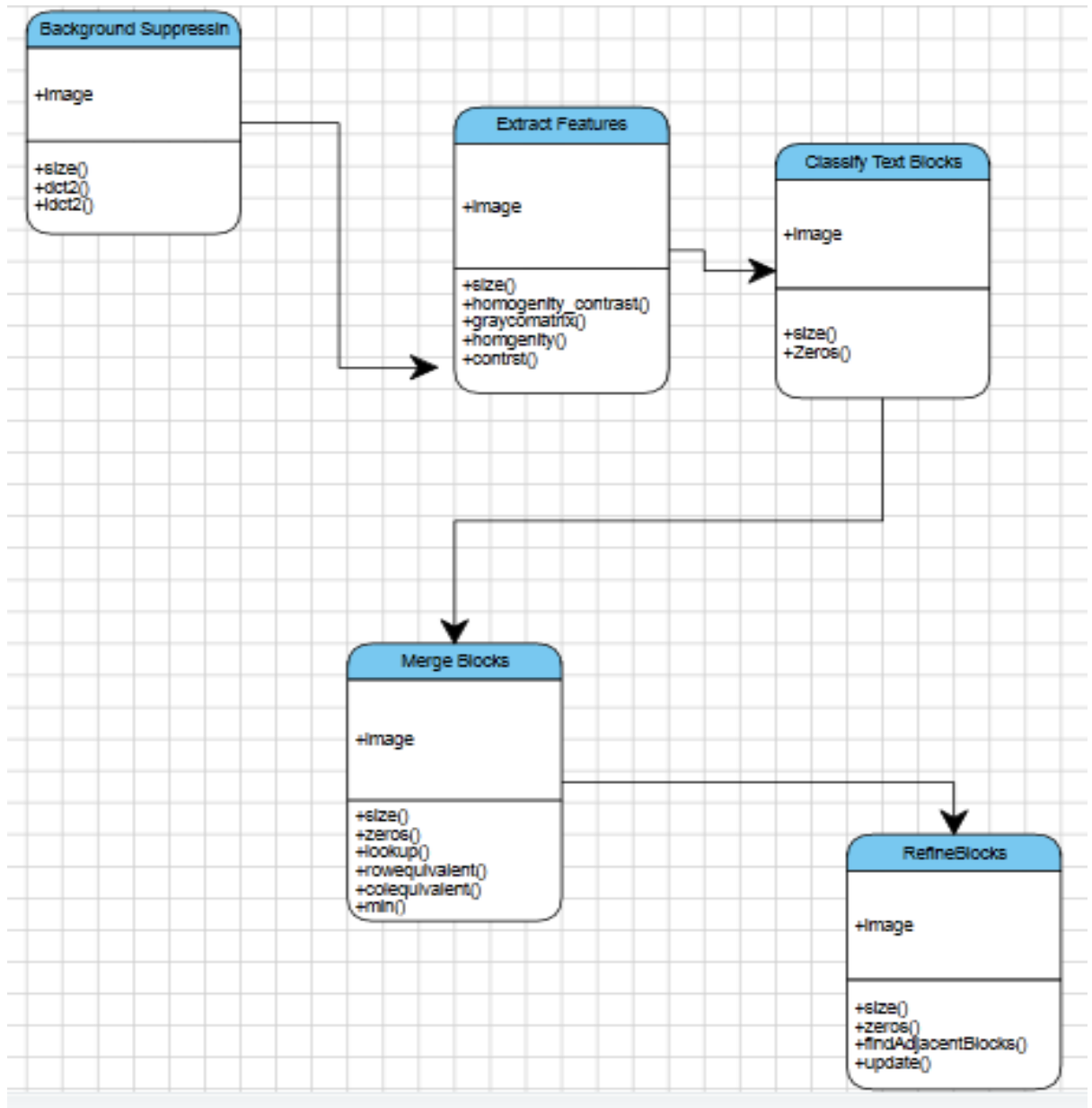


Figure3.3 Class Diagram

### 3.6.3 Sequence Diagram

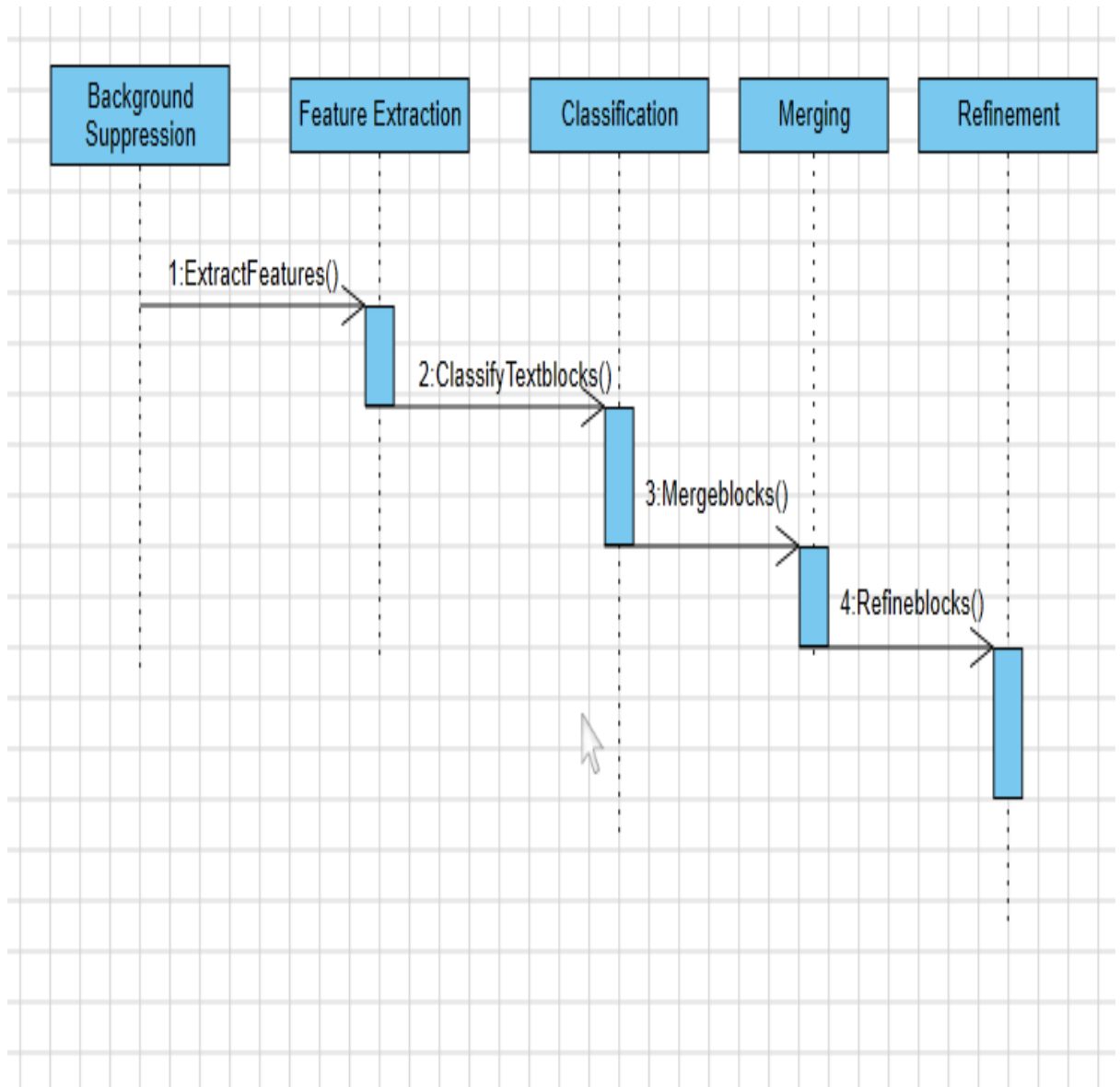


Figure 3.4 Sequence Diagram



## **4. REQUIREMENTS SPECIFICATION**

### **4.1 Software Requirements**

Operating System: Microsoft Windows (XP professional and the next versions)

**Tool:** Matlab Version: R2008a

Image Processing Toolbox

Neural Network Toolbox

### **4.2 Hardware Requirements**

Hard disk: 240 GB

Memory (RAM): 3.00 GB

Processor: intel®Core™Duo CPU @2.00GHz

## 5. IMPLEMENTATION

We made very few modifications to the above algorithm so that it can detect the text regions in which text is written in Telugu, English and Hindi. The texture features are dependent on the image, means some images are highly contrasted whereas some images are less contrasted. To deal with the problem, we've considered the threshold as function of average contrast and homogeneity instead of taking constant threshold. And the refinement process has been changed slightly. We have observed that if refinement process is carried before the merging step, we can improve the detection rate but the system has higher false positive rate. Thus, we avoid refinement step. The proposed method is as follows:

### 5.1 Background removal/suppression in the DCT domain

The constant background is removed successfully by applying a high pass filter that attenuates the DC component of every 8x8 DCT block of the image  $f(x, y)$  of size  $L \times M$ , where  $x$  and  $y$  are spatial coordinates. The transform function of high pass filter that operates on every 8x8 DCT block is given in equation 1. Later the processed/background suppressed image  $g(x, y)$  is obtained by applying inverse DCT on every 8x8 DCT block, which will be used in subsequent phases. The steps of background suppression are depicted in equations 2, 3 and 4. The block diagram of background suppression using DCT is given in figure 2.

$$H(u, v) = \begin{cases} 0, & \text{if } (u, v) = (1, 1) \\ 1 & \text{otherwise} \end{cases} \dots\dots\dots(5.1)$$

The high pass filter attenuates the DC component by storing value zero in every top left coordinate of 8x8 DCT block. This process is also called removing low frequency component from top left corner from every DCT block.

$$G(u, v) = DCT[f(x, y)] \dots\dots\dots(5.2)$$

where  $1 \leq x, u \leq L$ , and  $1 \leq y, v \leq M$

$$P(u, v) = H(u, v) G(u, v) \dots\dots\dots(5.3)$$

$$g(x, y) = DCT^{-1}[P(u, v)] \dots\dots\dots(5.4)$$

Where,

$G(u,v)$  is DCT matrix of input image  $f(x,y)$ .

$P(u,v)$  is Processed DCT matrix.

$g(x,y)$  is background suppressed image

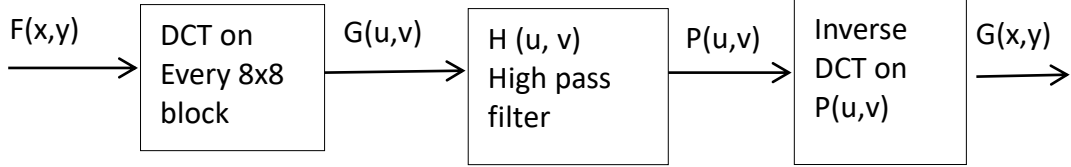


Figure 5.1 High Pass Filter for Background Removal using DCT

## 5.2 Features Extraction

In this phase, the *texture* features such as *homogeneity* and *contrast* are obtained from every 50x50 block of the processed image  $g(x,y)$  at  $0^\circ, 45^\circ, 90^\circ$ , and  $135^\circ$  orientations. Totally 8 features are extracted from every block and are stored into a feature vector  $X_i$  (Subscript “i” corresponds to  $i^{\text{th}}$  block). The feature vector  $X_i$  also records block coordinates which corresponds to minimum and maximum row and column numbers of the block. Feature vectors of all  $N$  blocks are combined to form a feature matrix  $D$  as depicted in equation 5. The feature vector is described in equation 6.

$$D = [X_1, X_2, X_3, \dots, X_N]^T \quad (5.5)$$

$$X_i = [rmin, rmax, cmin, cmax, f_1, \dots, f_8] \quad (5.6)$$

Where,

$rmin, rmax, cmin, cmax$  corresponds to coordinates of  $i^{\text{th}}$  block in terms of minimum and maximum row and column numbers.

$f_1$  and  $f_2$  corresponds to homogeneity and contrast at 0 degree orientation.

$f_3$  and  $f_4$  corresponds to homogeneity and contrast at 45 degree orientation.

$f_5$  and  $f_6$  corresponds to homogeneity and contrast at 90 degree orientation.

$f_7$  and  $f_8$  corresponds to homogeneity and contrast at 135 degree orientation.

The features *homogeneity* and *contrast* are calculated as in equations 7 and 8

$$\text{Homogeneity} = \sum_{i=1}^Q \sum_{j=1}^Q \left( \frac{p(i,j)}{R} \right)^2 \quad (5.7)$$

$$\text{Contrast} = \frac{\sum_{i=1}^Q \sum_{j=1}^Q \sum_{|i-j|=n} p(i,j)}{N} \quad (5.8)$$

$$R = \frac{\sum_{i=1}^Q \sum_{j=1}^Q p(i,j)}{N} \quad (5.9)$$

Where,

$P$  corresponds to cooccurrence matrix at a given degree.

$R$  is normalized value of cooccurrence matrix  $P$ .

$N$  is total number of blocks.

$Q \times Q$  is dimension of block size which is chosen as  $50 \times 50$ .

### 5.3 Classification

The classification phase of the proposed model uses discriminant functions to classify every block into two classes'  $w_1$  and  $w_2$  based on feature vector  $X_i$ . Where,  $w_1$  corresponds to text blocks and  $w_2$  corresponds to non-text blocks category. The discriminant functions uses two thresholds  $T_1$  and  $T_2$  corresponding to *homogeneity* and *contrast* values respectively. The discriminant functions  $d_1$  and  $d_2$  together decides a block as text block if the *homogeneity* and *contrast* features values at  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$  and  $135^\circ$  orientations are less than thresholds  $T_1$  and  $T_2$ .

The classification rules using discriminant functions are stated in equations 10 and 11. Given a feature matrix  $D$  of features  $X_i$ , assign the corresponding image block to Class  $w_1$  if  $d_1(X_i)$  is satisfied, and  $d_2(X_i)$  is satisfied

Class  $w_2$  otherwise

Where,

$d_1(X_i)$  is a discriminant function which defines/specifies constraint on homogeneity value.

$d_2(X_i)$  is a discriminant function which defines/specifies constraint on Contrast value.

$$d_1(X_i) = \begin{cases} \text{satisfied if } X_i(f_j) \leq T_1, \forall i=1, N \text{ and } j = 1, 3, 5, 7 \\ \text{Not satisfied otherwise} \end{cases} \quad (5.10)$$

$$d_2(X_i) = \begin{cases} \text{satisfied if } X_i(f_j) \geq T_2 \forall i=1, N \text{ and } j = 2, 4, 6, 8 \\ \text{Not satisfied otherwise} \end{cases} \quad (5.11)$$

Where,

$T_1$  corresponds to threshold on homogeneity

$T_2$  corresponds to threshold on contrast.

These two values are calculated as average values of homogeneity and contrast in all orientations

The coordinates  $rmin$ ,  $rmax$ ,  $cmin$ ,  $cmax$  which corresponds to minimum and maximum row and column numbers of every classified text block  $C_i$  will be stored into a new vector  $B$  as in equation 12, which are later used during merging process to obtain contiguous text regions.

$$B = [C_i, i = 1, N_1] \dots \dots \dots (5.12)$$

$$C_i = [rmin, rmax, cmin, cmax] \dots \dots \dots (5.13)$$

Where,

$C_i$  corresponds to  $i^{th}$  text block.

$rmin$ ,  $rmax$ ,  $cmin$ ,  $cmax$  corresponds to the coordinates of  $i^{th}$  block in terms of minimum and maximum row and column numbers.

$N_1$  is Number of text blocks.

#### 5.4 Merging of text blocks to detect text regions

The merging process combines the potential text blocks  $C_i$ , connected in rows and columns, to obtain new text regions  $r_i$ , whose coordinates are recorded into vector  $R$  as depicted in equation 14.

$$R = [r_i, i = 1, W] \dots \dots \dots (5.14)$$

$$r_i = [rmin, rmax, cmin, cmax] \dots \dots \dots (5.15)$$

where,

$r_i$  corresponds to  $i^{th}$  text region

$rmin$ ,  $rmax$ ,  $cmin$ ,  $cmax$  corresponds to the coordinates of  $i^{th}$  text region.

$W$  is number of text regions.

The merging procedure is described in algorithm 1;

#### Algorithm1

**Input:** Vector  $B$  which contains coordinates of identified text blocks

**Output:** Vector  $\mathbf{R}$  which records text regions

**Begin**

1. Choose the first block  $C_s$  from vector  $\mathbf{B}$ .
2. Initialize coordinates of a new text region  $r_i$  to coordinates of **block**  $C_s$ .
3. Select next block  $C_p$  from the vector  $\mathbf{B}$ .
4. **if** (the block  $C_p$  is connected to  $r_i$  in row or column) **then**  
    **begin**  
        Merge and update coordinates  $rmin, rmax, cmin, cmax$  of block  $r_i$ .  
         $rmin = \min\{ r_i[rmin], C_p[rmin] \}$   $rmax = \max\{ r_i[rmax], C_p[rmax] \}$   
         $cmin = \min\{ r_i[cmin], C_p[cmin] \}$   $cmax = \max\{ r_i[cmax], C_p[cmax] \}$   
    **else**  
        Store text region  $r_i$  into vector  $\mathbf{R}$ .  
        Initialize coordinates of a new text region  $r_i$  to coordinates of current **block**  $C_p$ .  
    **end**
5. Repeat steps 2 -5 until  $p=N_1$ .

**End**

### 5.5 Refinement of text regions

The refinement phase is a post processing step used to improve the detection accuracy. This phase is concerned with refining the size of the detected text regions to cover small portions of missed text present in adjacent undetected blocks and unprocessed regions. The refinement process is carried out in two steps; Adjacent undetected blocks processing and Combining unprocessed region. The functionality of each step is described below:

#### 5.5.1 Adjacent undetected blocks processing

In this step, every detected text region  $r_i$  is refined either by combining the entire size (50 rows and 50 columns) or selected rows and columns of adjacent undetected blocks (in rows and columns) which contain/cover small portions of missed text. The *average contrast* value  $(f_2+f_4+f_6+f_8) / 4$  is computed for every adjacent undetected block (in row or column), If the computed value is greater than or equal to 35 then the entire block size is combined with region  $r_i$  for refinement assuming that the missed adjacent block may contain significant text information. The heuristic value 35 is chosen empirically based on experiments. Similarly, if the *average contrast* value is between 5-9, 10-19, and 20-34, then 5, 10, and 20 respective adjacent rows/columns are added to region  $r_i$  for refinement. Again the heuristic values 5, 10 and 20 are chosen empirically based on experiments and results were encouraging.

### 5.5.2 Combining unprocessed regions

The presented method works by dividing an image of size 240x320 into 50x50 sized blocks till the completion of phase 3.3. Hence the remaining 40 rows and 20 columns will be left unprocessed after phase 3.3. As the unprocessed rows and columns may also contain text information they need to be processed for further refinement of detected text regions. In this step, the detected text regions are further refined by processing and adding adjacent unprocessed area to cover missed portion containing small text. During processing only *average contrast* feature value  $(f_2+f_4+f_6+f_8) / 4$  is computed from unprocessed area, and if the *average contrast* value is greater than or equal to the threshold then the entire size of the unprocessed region is combining with adjacent detected text regions for refinement. The proposed methodology is robust and performs well for different sizes of font and image resolution. The block size is an important design parameter whose dimension must be properly chosen to make the method more robust and insensitive to variation in size, font and its alignment. The approach also detects nonlinear text regions and results are presented in the next section. However, the method detects larger text regions than the actual size when the image background is more complex containing trees, vehicles, and other details from sources of outdoor scenes. The method takes about 6 to 10 seconds of processing time based on the complexity of background contained in the image.

The procedure for merging is described in algorithm 2.

#### Algorithm 2

**Input:** - Vector **R** which contains coordinates of extracted text regions.

- Vector **D** which contains texture features and coordinates of all 24 blocks of preprocessed image.

- Unprocessed 40 rows and 20 columns of the preprocessed image

**Output:** Vector **R** which records refined text regions.

#### Begin

1. Start with first text region  $r_i$ .
2. Select the next feature vector  $X_p$  from the feature matrix **D**.
3. **if** (the feature vector  $X_p$  is connected to text region  $r_i$  in row or column) **then**  
    **begin**
  - 3.1 Find average *contrast* value =  $(f_2+f_4+f_6+f_8) / 4$ .
  - 3.2 **if** (average *contrast* value is  $> 35$ ) **then**

```

begin
  Merge and update coordinates rmin, rmax, cmin, cmax of text region ri by
  adding adjacent block.
  rmin = min{ ri[rmin ], Cp[rmin] } rmax = max{ ri[rmax ], Cp[rmax] }
  cmin = min{ ri[cmin ], Cp[cmin] } cmax = max{ ri[cmax ], Cp[cmax] }
else
if (average contrast value is between 5 to 9 ) then
  begin
    cmin = cmin - 5; // add 5 columns if feature vector Xp left connected
    or
    cmax = cmax + 5; // add 5 columns if feature vector Xp right connected
    or
    rmin = rmin - 5; // add 5 rows if feature vector Xp top connected
    or
    rmax = rmax + 5; // add 5 rows if feature vector Xp bottom connected
  end
if (average contrast value is between 10 to 19 ) then
  begin
    cmin = cmin - 10; // add 5 columns if feature vector Xp left connected
    or
    cmax = cmax + 10; // add 5 columns if feature vector Xp right connected
    or
    rmin = rmin - 10; // add 5 rows if feature vector Xp top connected
    or
    rmax = rmax + 10; // add 5 rows if feature vector Xp bottom connected
  end
if (average contrast value is between 20 to 34 ) then
  begin
    cmin = cmin - 20; // add 5 columns if feature vector Xp left connected
    or
    cmax = cmax + 20; // add 5 columns if feature vector Xp right connected
    or
    rmin = rmin - 20; // add 5 rows if feature vector Xp top connected
    or
    rmax = rmax + 20; // add 5 rows if feature vector Xp bottom connected
  end
end
4. Repeat steps 2 -3 until p = Nl.
5. Select next text region ri = [rmin, rmax, cmin, cmax] from R.
6. Repeat steps 2-5 until all text regions are refined.
End

```



## 6. RESULTS AND OBSERVATIONS

### 6.1 RESULTS

The proposed methodology for text region detection and extraction has been evaluated for 20 low resolution natural scene display board images using with complex backgrounds. The experimental tests were conducted for most of the images containing telugu, hindi text and few containing English text and results were highly encouraging. The experimental results of processing a typical display board image containing text with varying background is described in section 6.1. And the results of processing several other display board images dealing with various issues and the overall performance of the system are reported in section 6.2.

#### 6.1 Text Region Extraction of a typical display board image

A display board image of size 240x320 given in figure 6.1a, containing text information having smaller and bigger telugu characters and complex backgrounds such as building walls, doors, and uneven lighting conditions is initially preprocessed to suppress the background using a high pass filter in the DCT domain. The experimental values of applying a high pass filter for the first 8x8 DCT block of image in figure 6.1 are given in Table 6.1.



a)Original Image



b) Background Suppressed image



c) Detected text regions before post processing



d) Refined text regions after processing

Figure 6.1 The image is extracted from the paper “Text Region Extraction from Low Resolution Natural Scene Images using Texture Features”

Table 6.1: Preprocessing of first 8x8 block of an image in figure 6.1 a

Step1: 8x8 first image block	Step2: 8x8 first DCT block
159 159 159 159 159 159 159 159 159 159 159 159 159 159 159 159 160 160 160 160 160 160 160 161 160 160 160 160 160 160 160 161 160 160 160 160 161 161 161 162 160 160 161 161 161 161 162 162 161 161 161 161 161 161 162 162 161 161 161 161 161 161 161 161	1.2821 -0.0019 0.0007 -0.0006 0.0004 -0.0002 0.0001 0.0002 -0.0062 0.0009 -0.0001 0.0001 0.0001 -0.0003 0.0002 -0.0002 -0.0015 0.0013 -0.0004 0.0003 -0.0004 0.0004 -0.0003 -0.0000 -0.0001 -0.0011 -0.0002 -0.0000 -0.0002 0.0003 -0.0001 0.0003 -0.0006 0.0003 -0.0001 0.0003 0.0001 -0.0004 0.0002 -0.0001 0.0007 0.0002 0.0004 -0.0003 0.0000 0.0002 -0.0001 -0.0001 0.0007 -0.0002 -0.0003 -0.0003 0.0000 0.0001 0.0001 0.0000 0.0009 -0.0003 0.0004 0.0002 0.0001 -0.0004 0.0000 -0.0000
Step 4: 8x8 first preprocessed image block after applying inverse DCT	Step3: 8x8 first DCT block after applying high pass filter
0 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 2 0 0 1 1 1 1 2 2 1 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1	0 -0.0019 0.0007 -0.0006 0.0004 -0.0002 0.0001 0.0002 -0.0062 0.0009 -0.0001 0.0001 0.0001 -0.0003 0.0002 -0.0002 -0.0015 0.0013 -0.0004 0.0003 -0.0004 0.0004 -0.0003 -0.0000 -0.0001 -0.0011 -0.0002 -0.0000 -0.0002 0.0003 -0.0001 0.0003 -0.0006 0.0003 -0.0001 0.0003 0.0001 -0.0004 0.0002 -0.0001 0.0007 0.0002 0.0004 -0.0003 0.0000 0.0002 -0.0001 -0.0001 0.0007 -0.0002 -0.0003 -0.0003 0.0000 0.0001 0.0001 0.0000 0.0009 -0.0003 0.0004 0.0002 0.0001 -0.0004 0.0000 -0.0000

The experimental values in Table 6.1 demonstrate that the constant gray level values in the range 159-162 in the first 8x8 block of the image in figure 6.1 have been compressed to a narrow range 0-2 after preprocessing. Hence, the transform function given in equation 1 that attenuates DC component as given in step3 has performed well in suppressing most of the constant background. The processed image after applying background suppression for all blocks is shown in figure 6.2 , where most of the unwanted details are removed. And only gray level discontinuities belonging to text and edges remain for further image analysis. Hence, the extracted texture features such as *homogeneity* and *contrast* from such preprocessed images aid classification decisions and help in increasing detection rate. All 24 50x50 image blocks of preprocessed image in figure 3b from which texture features are extracted are shown in figure 6.2. Table 6.2 summarizes the extracted features.

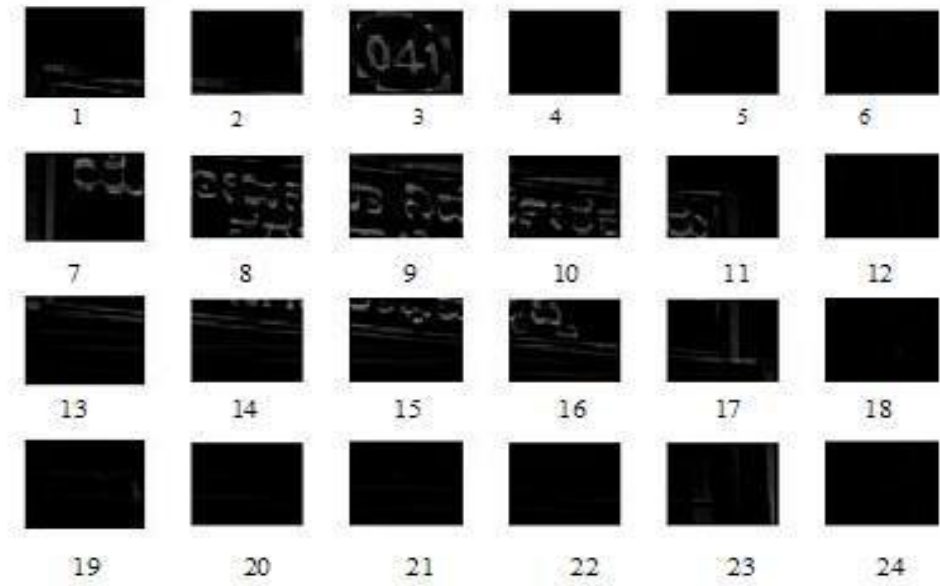


Figure 6.2: 24 50x50 blocks of preprocessed image in figure 6.1b.

Table 6.2: Feature Matrix D containing extracted features of image of size 240x320 given in figure 6.1 b

block	r min	rmax	cmin	cmax	F1	F2	F3	F4	F5	F6	F7	F8
1	1	50	1	50	0.35563	5.7518	0.27407	40.044	0.29799	33.25	0.27741	33.556
2	1	50	51	100	0.37939	3.9122	0.30362	25.075	0.32683	21.255	0.31086	22.454
3	1	50	101	150	<b>0.23571</b>	<b>145.05</b>	<b>0.21203</b>	<b>256.51</b>	<b>0.26159</b>	<b>115.69</b>	<b>0.21793</b>	<b>224.92</b>
4	1	50	151	200	0.5486	0.060408	0.4624	0.14452	0.49457	0.11714	0.46736	0.13869
5	1	50	201	250	0.55128	0.056735	0.47864	0.11308	0.51388	0.081837	0.48201	0.11058
6	1	50	251	300	0.41841	0.21367	0.35536	0.33944	0.40281	0.19714	0.35614	0.33882
7	51	100	1	50	<b>0.23662</b>	<b>80.981</b>	<b>0.20995</b>	<b>134.79</b>	<b>0.26495</b>	<b>9.38</b>	<b>0.21274</b>	<b>121.14</b>
8	51	100	51	100	<b>0.29437</b>	<b>108.49</b>	<b>0.23146</b>	<b>266.22</b>	<b>0.27296</b>	<b>167.09</b>	<b>0.24404</b>	<b>213.76</b>
9	51	100	101	150	<b>0.27832</b>	<b>94.329</b>	<b>0.21101</b>	<b>271.21</b>	<b>0.24637</b>	<b>182.34</b>	<b>0.23078</b>	<b>220.13</b>
10	51	100	151	200	<b>0.26405</b>	<b>70.58</b>	<b>0.1898</b>	<b>185.71</b>	<b>0.22773</b>	<b>130.81</b>	<b>0.20226</b>	<b>169.87</b>
11	51	100	201	250	<b>0.35865</b>	<b>31.801</b>	<b>0.29939</b>	<b>78.564</b>	<b>0.35312</b>	<b>47.617</b>	<b>0.31177</b>	<b>60.969</b>
12	51	100	251	300	0.45093	0.17143	0.38476	0.26801	0.4281	0.1451	0.37781	0.27239
13	101	150	1	50	0.31607	5.8076	0.21355	52.554	0.2322	44.147	0.22656	42.147
14	101	150	51	100	<b>0.29954</b>	<b>31.962</b>	<b>0.19948</b>	<b>90.116</b>	<b>0.21792</b>	<b>57.519</b>	<b>0.2132</b>	<b>71.612</b>
15	101	150	101	150	<b>0.29219</b>	<b>48.188</b>	<b>0.19208</b>	<b>171.58</b>	<b>0.2114</b>	<b>135.96</b>	<b>0.20367</b>	<b>154</b>
16	101	150	151	200	<b>0.2646</b>	<b>50.764</b>	<b>0.16705</b>	<b>134.08</b>	<b>0.19386</b>	<b>90.476</b>	<b>0.18203</b>	<b>118.49</b>
17	101	150	201	250	0.30584	18.546	0.24364	49.318	0.30325	29.852	0.25069	43.421
18	101	150	251	300	0.47275	0.26082	0.42262	0.45981	0.4802	0.2702	0.43067	0.41733
19	151	200	1	50	0.38193	2.4163	0.32077	3.747	0.35694	1.4843	0.32784	3.3561
20	151	200	51	100	0.4088	0.095102	0.32114	0.87568	0.33529	0.80796	0.32327	0.8284
21	151	200	101	150	0.37253	0.22653	0.2801	1.3925	0.2939	1.2329	0.28104	1.3711
22	151	200	151	200	0.42539	0.073673	0.31561	1.3961	0.32448	1.3386	0.31156	1.3282
23	151	200	201	250	0.22828	48.116	0.19393	53.1	0.26798	3.7524	0.19778	49.633
24	151	200	251	300	0.52383	0.15612	0.48039	0.2045	0.5277	0.083878	0.48225	0.19992

The experimental values in Table 2 demonstrate the values of coordinates and texture features extracted at 0, 45, 90 and 130 degree orientations of all 24 blocks shown in figure 6.2. As per figure 3e only 9 blocks numbered 3, 7-11, and 14-16 are text blocks and remaining blocks are non-text blocks. The classification phase recognizes 6 blocks numbered 3, 7-10 and 16 as text blocks as their *homogeneity* and *contrast* values satisfy discriminant functions given in equations 10 and 11. And their coordinate values are recorded into feature vector  $B$  as shown in Table 3. The blocks 11, 14, and 15 which contain small text does not satisfy discriminant functions and are not recognized as text blocks. But they are handled properly during merging and post processing phases of the proposed method to improve the system performance.

Table 6. 3: Vector  $B$  showing coordinate values of identified text blocks

$rmin$	$rmax$	$cmin$	$cmax$
1	50	101	150
151	100	1	50
151	100	51	100
151	100	101	150
151	100	151	200
101	150	151	200

The methodology merges identified text blocks and detects a single text region whose coordinates are stored into vector  $R$  as shown below. The detected text region also covers blocks 14 and 15 during merging which were undetected after classification phase thus improving detection accuracy. And the corresponding extracted text region of the image is shown in figure 6.1.c.

$$R = [1 \ 150 \ 1 \ 200]$$

Where  $rmin = 1$ ,  $rmax = 150$

$cmin = 1$ ,  $cmax = 200$

The extracted text region now undergoes post processing phase, where during step1, the adjacent blocks 11 and 17 in right column are combined with the detected text region as their average *contrast* values satisfy the first condition in *step 3.2* of *algorithm2*. It is also noted that the non-text block 17 is false accepted. And during

step2, the adjacent unprocessed areas are not combined as they do not contain text information. Hence, the final refined text region coordinates are now shown below;

$R = [1 \ 150 \ 1 \ 250]$   
Where  $r_{min} = 1$ ,  $r_{max} = 150$   
 $c_{min} = 1$ ,  $c_{max} = 250$

The system performance after carrying out all the phases on the image shown in figure 6.2 is evaluated as 66.6 % where out of 9 text blocks in the total 24 blocks 6 text blocks are correctly detected and 3 blocks are missed.

### **6.1.1 Text Region Extraction: An experimental analysis dealing with various issues**

The text detection and extraction results of testing several different low resolution natural scene display board images dealing with various issues are shown in figures 6.3. And the corresponding detailed analysis is presented in Table 6.5.

The output of the system described in Table 6.4 brings out the following details after post processing;

**Detection Rate** = (Number of text blocks correctly detected/ Number of text blocks tested) \* 100 =  $(9/9) * 100 = 100\%$

**False Acceptance Rate (FAR)** = (Number of text blocks falsy detected/ Number of text blocks tested) \* 100 =  $(01/09)*100 = 11\%$

**False Rejection Rate (FRR)** = (Number of missed text blocks / Number of text blocks tested) \* 100

Serial Number	a) Original images	b) Background Suppressed image	c) Detected text blocks after classification	d) Text regions after merging
1				
2				
3				
4				
5				
6				

Figure 6.3: Text Extraction results of processing 6 low resolution natural scene images dealing with various issues

Table 6.5: The performance of the system of processing different images given in figures 4-5 dealing with various issues

Input Image	No of text blocks	Correctly detected text blocks	Falsely detected text blocks	Missed text blocks	Detection rate%
1	10	10	00	00	100
2	11	11	03	00	100
3	14	14	02	00	100
4	10	10	01	00	100
5	12	10	03	02	83
6	12	11	00	01	91.6

The present methodology has produced good results for natural scene images containing text of different size, font, and alignment with varying background. The approach also detects nonlinear text regions. The method is advantageous as it uses only two texture features for text extraction. The advantage lies in less computation involved in feature extraction and classification phases of the method. The reason for false reject rate is the low contrast energy of blocks containing minute part of the text, which is too weak for acceptance to classify blocks as text blocks. However, the method has detected larger region than the actual size of the text region, when display board images with more complex backgrounds containing trees, buildings and vehicles are tested. The system is developed in MATLAB and evaluated for 20 low resolution natural scene images on Intel Celeron (1.4GHz) computer. And it was observed that the processing time lies in the range of 6 to 10 seconds due to varying background.

Overall system performance is evaluated to 89.2% where this approach is tested with 20 different images. The total text blocks in the 20 images is summed up to 224 blocks, 200 text blocks are correctly detected, 40 blocks are falsely detected and 24 blocks are missed

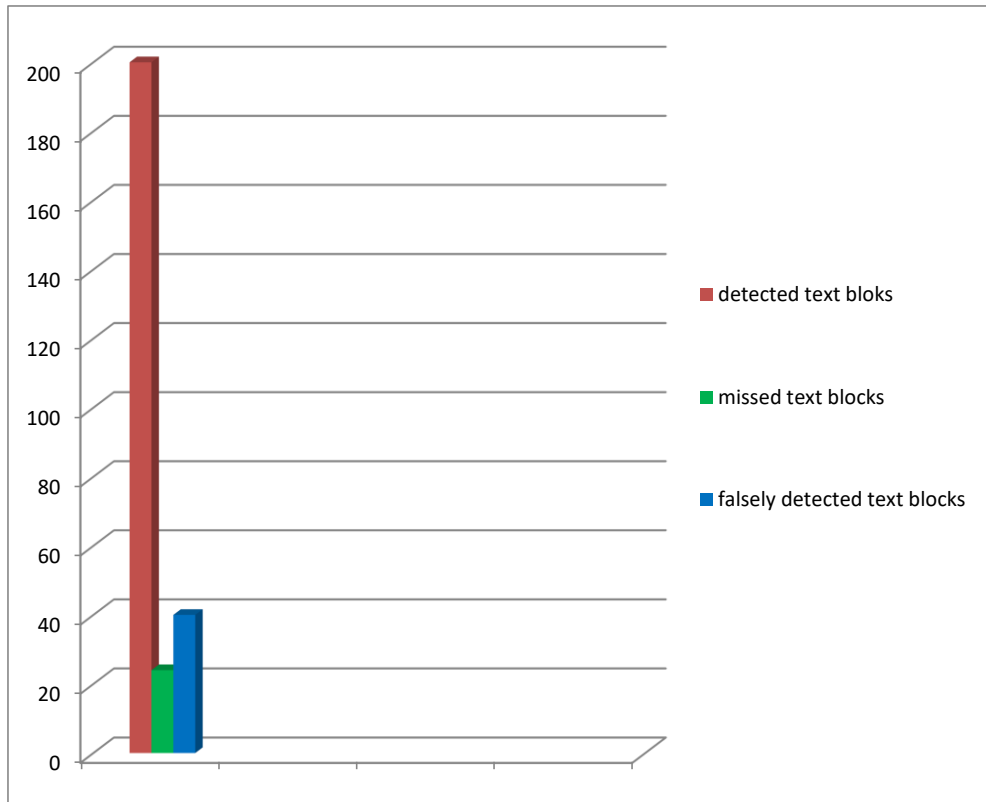


Figure 6.4 Overall results of proposed model for text localization

## 6.2 Observations

### 6.2.1 Performance evaluation:

In order to test the robustness and performance of the each algorithm, The algorithm was tested on the image data set of 20 low resolution images The precision and recall rates (Equations (1) and (2)), have been computed based on the number of correctly detected blocks in an image in order to further evaluate the efficiency and robustness of the method.

The Precision rate is defined as the ratio of correctly detected blocks to the sum of correctly detected blocks plus false positives. *False positives* are those regions in the image which are actually not characters of a text, but have been detected by the algorithm as text regions.



$$\text{Precision rate} = \frac{\text{Correctly detected blocks}}{\text{Correctly detected blocks} + \text{False positives}} \times 100\% \dots\dots\dots (6.1)$$

The Recall rate is defined as the ratio of correctly detected blocks to the sum of correctly detected blocks plus false negatives. *False Negatives* are those regions in the image which are actually text characters, but have not been detected by the algorithm.

$$\text{Recall rate} = \frac{\text{Correctly detected blocks}}{\text{Correctly detected blocks} + \text{False negatives}} \times 100\% \dots\dots\dots (6.2)$$

We found that the system has **50% precision** and **77%** recall without refinement step.

We also observed that with refinement of blocks, the system got **47% precision** and **93% recall**.

Table 6.7 Precision and Recall rates of the overall system

	Precision	Recall
With Refinement	50	77
Without Refinement	47	93

From the above table, it's clear that refinement process increasing the detection rate but it has so many false positives.

### 6.2.1 Evaluation of Text detection methods:

The texture based approach which is implemented as a part of this project is compared with the already existing methods Edge based method [2, 3] and connected component based method [4]. Codes for the other two methods are available online .we have run the codes and tested the methods on the same dataset that we have used to test the texture based method. The results of the three methods on the same dataset are shown in the following table.

Serial number	a)Original image	b Edge based approach	c)Connected component based approach	d)Texture based approach
1				
2				
3				
4				
5				
6				
7				

Figure 6.5: Result Images of Three detection Methods

We compare the performance of the texture based method with the edge based and connected component method based on the results obtained on implementing each method and testing on the same dataset. The results for precision and recall rates and average run time obtained by each method when tested on the 20 images in the dataset have been listed in Table 6.8

Table 6.8 Comparison of the three methods on the common test set.

Method	Precision Rate	Recall rate	Average Runtime (seconds)
Texture based method (without refinement)	47	93	7
Edge based method	47.4	75.09	15
Connected component based method	50.10	73.42	19

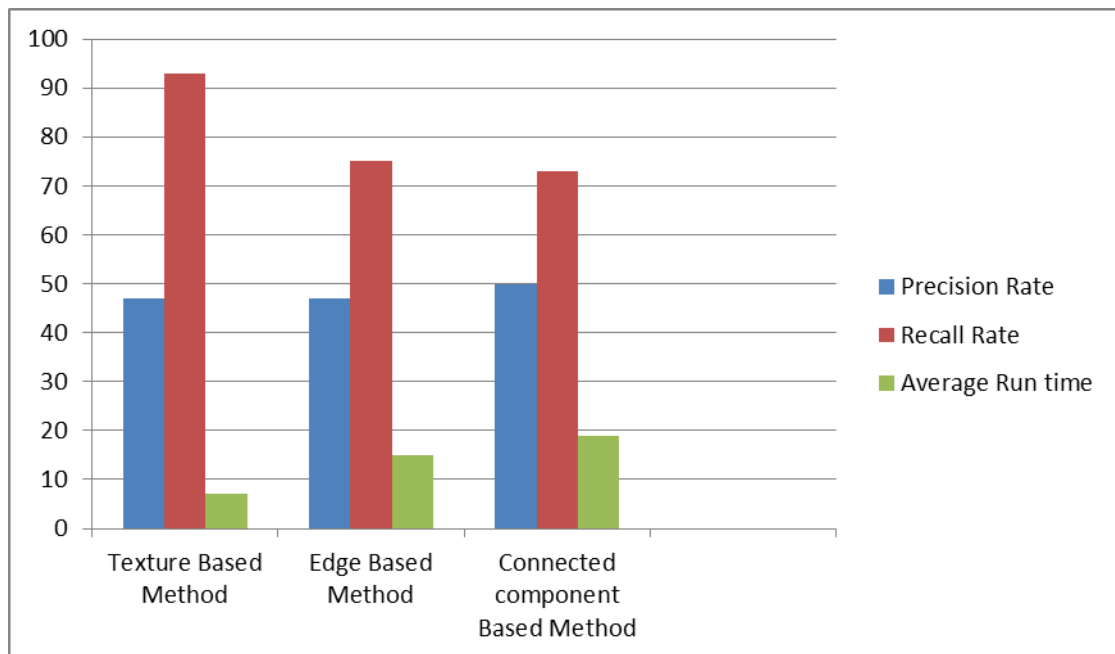


Figure 6.6 Performance Measure of three Methods

The graph in Figure 6.7 shows that the average precision rate obtained by the connected component (50.10%), The edge based algorithm (47.4%) and Texture based algorithm (47) are very close to each other. The average recall rates obtained by the texture based algorithm (93%) is higher than those obtained by the connected component algorithm (74.32%) and edge based algorithm (75.09). The average run time of Texture based method (7 sec ) is very fast in giving results compared to edge based method (15 sec) and texture based method (19 sec).

## **7. CONCLUSIONS AND FUTURE SCOPE**

### **7.1 Conclusions**

The implemented system detects text regions with an overall Precision (47 %) Recall (93 %) and average run time (7 sec) is more efficient compared to that of the performance obtained with edge based method and connected component based method. As the method is based on texture features, if the image contains large trees, vehicles, etc., it may detect them as text regions. So the system doesn't have high precision as so many images in the dataset, contain large trees which has high texture values. We observed that none of the methods, texture based and Edge detection based methods are not good enough to detect the text regions. One possible way to reduce the false positives is, if we apply edge detection based method in the detected regions, we may reduce the false positives.

### **7.2 Future Scope**

For future work the following recommendations can be taken into consideration:

1. Combining the edge, connected component based and texture based algorithms: Each of the algorithms is by itself quite robust in extracting text regions from natural images. A combination of these techniques can produce more efficient outputs.
2. Morphological cleaning of images: The approach used by the edge based as well as the connected component based algorithm does not take into consideration the removal of noise or unwanted clutter from the test images before or after the computations. A morphological cleaning operation would be helpful in reducing the number of false positives obtained. Thus, cleaning of the image could result in a higher precision rate.
3. Testing on different kind of images: The goal of this project was to implement a texture based method for text detection in natural images proposed in [1]. In order to more thoroughly evaluate this technique, the algorithm can be tested on different kind

of images and video frames such as movie clips, animated scenes, comic book images, as well as document images.

4. Testing for hidden text region extraction in a cluttered scene: An interesting test would be to find text regions which are hidden behind other objects or water marked within an image. In order to achieve this, various other approaches mentioned in the Earlier sections can be explored

## REFERENCES

- [ 1] Angadi, S.A. and Kodabagi, M.M, *Text region extraction from low resolution natural scene images using texture features*, Advance Computing Conference (IACC), 2010 IEEE 2ndInternational
- [ 2] Xiaoqing Liu and Jagath Samarabandu, *An Edge-based text region extraction algorithm for Indoor mobile robot navigation*, Proceedings of the IEEE, July 2005.
- [ 3] Xiaoqing Liu and Jagath Samarabandu, *Multiscale edge-based Text extraction from Complex images*, IEEE, 2006.
- [ 4] Julinda Gllavata, Ralph Ewerth and Bernd Freisleben, *A Robust algorithm for Text Detection in images*, Proceedings of the 3rd international symposium on Image and Signal Processing and Analysis, 2003.
- [ 5] Anoual, H. and Aboutajdine, D. and Elfkihi, S. and Jilbab, A, *Features extraction for text detection and localization*, I/V Communications and Mobile Network (ISVC)
- [ 6] Sushma, J. and Padmaja, M. Intelligent Agent & Multi-Agent Systems, *Text detection in color images*, 2009, IAMA 2009
- [ 7] Partio, M. and Cramariuc, B.and Gabbouj, M. and Visa, A, *Rock texture retrieval using gray level co-occurrence matrix*, Proc. of 5th Nordic Signal Processing Symposium
- [ 8] Shehzad Muhammad Hanif and Lionel Prevost, *Texture based Text Detection in Natural Scene Image: A help to blind and visually impaired persons*, Conference & Workshop on Assistive Technologies for People with Vision & Hearing Impairments Assistive Technology for All Ages CVHI 2007, M.A. Hersh (ed.)
- [ 9] C.P. SUMATHI, T. SANTHANAM and N.PRIYA, *Techniques and Challenges of automatic Text Extraction in Complex Images: A Survey*, Journal of Theoretical and Applied Information Technology, january 2012.
- [ 10] Xiaoqing Liu and Jagath Samarabandu, *A Simple and Fast Text Localization Algorithm for Indoor Mobile Robot Navigation*, Proceedings of SPIE-IS&T Electronic Imaging, SPIE Vol. 5672, 2005.
- [ 11] Qixiang Ye, Qingming Huang, Wen Gao and Debin Zhao, *Fast and Robust text detection in images and video frames*, Image and Vision Computing 23, 2005.
- [ 12] Qixiang Ye, Wen Gao, Weiqiang Wang and Wei Zeng, *A Robust Text Detection Algorithm in Images and Video Frames*, IEEE, 2003.
- [ 13] Tollmar K. Yeh T. and Darrell T. “*IDEixis - Image-Based Deixis for Finding Location-Based Information*”, In Proceedings of Conference on Human Factors in Computing Systems (CHI’04), pp.781-782, 2004.
- [ 14] Yu Zhong, Kalle Karu, and Anil. K. Jain. “*Locating Text in Complex Color Images*”, Pattern Recognition, 28(10):1523-1535, 1995.
- [ 15] V. Wu, R. Manmatha, and E. M. Riseman. “*TextFinder: An Automatic System to Detectand Recognize Text in Images*”, IEEE Transactions on Pattern Analysis and MachineIntelligence, 21(11):1224-1229, 1999.
- [ 16] V. Wu, R. Manmatha, and E. R. Riseman. “*Finding Text in Images*”, In Proceedings of ACM International Conference on Digital Libraries, pp. 1-10, 1997
- [ 17] H. Li. D. Doerman and O. Kia. “*Automatic Text Detection and Tracking in Digital Video*”,IEEE Transactions on Image Processing, 9(1):147-156, January 2000.

## **APPENDIX**

### **MATLAB**

The tool used for the development of the Identification system is MATLAB 7.9.0.529 R2009b version. The name MATLAB stands for Matrix Laboratory.

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. It allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non interactive language such as C or Fortran.

The MATLAB high-performance language for technical computing integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

Typical uses include

- Math and computation
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of add-on application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions that extend the MATLAB environment to solve particular classes of problems.



## **The MATLAB System**

The MATLAB system consists of these main parts:

### **Desktop Tools and Development Environment**

This part of MATLAB is the set of tools and facilities that help you use and become more productive with MATLAB functions and files. Many of these tools are graphical user interfaces. It includes: the MATLAB desktop and Command Window, an editor and debugger, a code analyzer, and browsers for viewing help, the workspace, and folders.

### **Mathematical Function Library**

This library is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigen values, Bessel functions, and fast Fourier transforms.

### **The Language**

The MATLAB language is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick programs you do not intend to reuse. You can also do "programming in the large" to create complex application programs intended for reuse.

### **Graphics**

MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully

customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

### **External Interfaces**

The external interfaces library allows you to write C and Fortran programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), for calling MATLAB as a computational engine, and for reading and writing MAT-files.

Among all the Toolboxes available, Image Processing Toolbox software is used.

### **Image Processing Toolbox**

The Image Processing Toolbox software is a collection of functions that extend the capability of the MATLAB numeric computing environment. The toolbox supports a wide range of image processing operations, including

- Spatial image transformations
- Morphological operations
- Neighborhood and block operations
- Linear filtering and filter design
- Transforms
- Image analysis and enhancement
- Image registration
- De-blurring
- Region of interest operations

Many of the toolbox functions are MATLAB M-files, a series of MATLAB statements that implement specialized image processing algorithms. MATLAB code for these functions is viewed using the statement - type function\_ name;

## TEST IMAGES

Input Image



Output Image

