International Journal of Document Analysis and Recognition (IJDAR) manuscript No.
(will be inserted by the editor)

Utpal Garain · Thierry Paquet · Laurent Heutte

On Foreground-Background Separation in Low Quality Document Images

Received: date / Accepted: date

Abstract This paper deals with effective separation of foreground and background in low quality document images suffering from various types of degradations including scanning noise, aging effects, uneven background or foreground, etc. The proposed algorithm shows an excellent adaptability to tackle with these problems of uneven illumination and local changes or non-uniformity in background and foreground colors. The approach is primarily designed for (not restricted to) processing of color documents but it works well in the gray scale domain too. Test document set considers samples (in color as well as in gray scale) of old historical documents including manuscripts of high importance. The data set used in this study consists of hundred images. These images are selected from different sources including image databases that had been scanned from working notebooks of famous writers who used to write with quill or pencil generating very low contrast between foreground and background. Evaluation of foreground extraction method has been judged by computing the accuracy of extracting handwritten lines and words from the test images. This evaluation shows that the proposed method can extract lines and words with accuracies of about 84% and 93%, respectively. Apart from this quantitative method, a qualitative evaluation is also presented to compare the proposed method with one popular technique for foreground/background separation in document images.

U. Garain
Computer Vision & Pattern Recognition Unit, Indian Statistical Institute, 203, B. T. Road, Kolkata 700108, INDIA
Tel.: +91-33-25752860
Fax: +91-33-25773035
E-mail: utpal@isical.ac.in

T. Paquet and L. Heutte
Laboratoire PSI - FRE CNRS 2645, UFR des Sciences, University of Rouen, 76821 Mont Saint Aignan cedex, FRANCE
Tel.:
Fax:
E-mail: {Thierry.Paquet,Laurent.Heutte}@univ-rouen.fr

1 Introduction

Binarization is considered as one of the important preprocessing steps in document image analysis (DIA) algorithms. This is so because compared to gray or color information the use of bi-level (foreground and background) information decreases the computational complexity and thereby enables the utilization of simplified analysis techniques. Therefore, in the field of document image analysis efficient binarization has been a subject of intense research during last several years.

A number of techniques have been proposed and found application mainly in the gray-scale domain as research in document processing has so far by and large been restricted to binary or gray-scale area only. On the other hand, with the widespread development of input devices for color images, documents like books, magazines, newspapers, personal notebooks, historical documents, etc. are now often stored into computers so that color information is preserved on the digital data as it is. Therefore research dealing with color documents has also gained considerable attention in the recent past.

With the introduction of color documents, binarization and foreground-background separation induce a subtle difference between them. Though binarization essentially does foreground-background separation in document images, but as far as color documents are concerned, we view that the later technique (i.e. foreground/background separation) refers to a more general aspect because color documents very often contains foreground elements (and may be background too) in different colors. Therefore, apart from labeling the image pixels as foreground or background, it seems to be more meaningful (and perhaps useful too in many DIA applications) if different labels (depending upon their color similarity) are maintained within different foreground (background) entities.

However, binarization methods proposed for gray-scale documents have not been well tested or extended for color documents. Instead, a very few studies that have been proposed for foreground extraction in color docu-

ments are quite different in nature from the traditional binarization methods and have been restricted to color domain only. In this sense, a gap is observed regarding the generality of the binarization techniques concerning the application domain (gray or color) and a more general approach is therefore called for.

Moreover, generality of an approach, in many cases, suffers regarding the type (printed or handwritten) of documents being processed. Though printed documents, in general, show a well contrasted background and foreground, such thing may not hold for many handwritten manuscripts. Furthermore, historical documents impose different types of degradations including aging effect, noise, uneven illumination, etc. Many of the existing approaches working in the gray-scale domain have addressed some of these issues but several other difficult situations still remain un-addressed. For example, in many handwritten manuscripts of historical/literary interest, text has been written with quill or pencil that sometimes does not generate very well contrasted foreground. Moreover, stroke marks are very often spread over the page background. Effective binarization in such cases still remain a challenging task.

This paper is aimed at addressing the above-mentioned issues and bridging the observed gaps within its capacity. Initially, it presents a general binarization techniques which is primarily aimed for (but not restricted to) color documents and applicable to gray-scale as well. The algorithm is designed for binarization of varieties of documents starting from images of well contrasted foreground and background to those suffered from many degradations like uneven illumination, noise, aging effect, etc. Moreover, the approach is applicable for printed as well as handwritten manuscripts. After binarization is achieved, the proposed technique attempts to locate different regions (based on color similarity) within the foreground part and give different labels to them and thereby helping other immediate DIA applications like page segmentation, text location, etc.

1.1 A brief survey of popular binarization and foreground-background separation techniques

As mentioned earlier that binarization because of its importance has been a subject of intense research interest during the last several years and summary of such techniques can be found in several papers like ones in [1]-[4]. These techniques have, so far, been applied for binarization of grey scale images but their potential for binarization of color documents has not been properly investigated.

A major commonality observed in these techniques is that most of them focus on one aspect of choosing threshold either globally or locally. The global threshold selection methods (e.g. [5,6]) assumes the gray level histogram is bimodal and then chooses a single threshold at valley point to label pixels into foreground or background classes. Experiments show that such a technique is simple and often effective too but breaks down when illumination, background or noise characteristics are non-uniform. As a remedy to these problems, local or adaptive thresholding schemes have been proposed. In local thresholding, threshold values are determined locally, e.g. pixel by pixel, or region by region.

In most cases, threshold is computed for every element (i.e. pixel or region) based on local statistics [7]. For example, the approach proposed by Niblack [9] attempts to vary the threshold over the image based on the local mean and standard deviation computed in a small neighborhood of each pixel. O'Gorman's method [10] chooses the threshold to optimize local connectivity whereas Tsai's method [8] tries choose threshold to preserve local low-order moments. Liu and Srihari [11] used the global Otsu [5] algorithm to obtain candidate thresholds. Then, texture features were measured from each thresholded image, based on which the best threshold was picked. Sauvola *et al.* classify page contents to background, pictures and text prior to apply different approaches to define threshold for each pixel. Recently, Gatos *et al.* [12] proposed another adaptive binarization technique for gray-scale images of low quality historical documents where Niblack's method [9] is initially applied to detect foreground parts but final binarization result is improved using several post-processing steps.

On the other hand, some binarization or foreground extraction techniques that have been proposed for color document images are broadly based on color clustering or color segmentation principle and in this sense, they are quite different from traditional threshold selection algorithms. However, despite the large number of proposed algorithms for color image segmentation [14] only a handful of them have found direct application for the document image processing. This is so because use of classical segmentation algorithm exhibit difficulties to tackle several document defects like stains, humidity marks, degradation of ink, paper, etc. Large size of color document images is another bottleneck for efficient use of the traditional segmentation strategies. Rather, generic algorithms, in few cases, have been customized in several ways for efficient background-foreground separation in color document images.

A few initial studies dealing with color documents concentrates on extraction of text parts. For example, studies by Lopresti and Zhou [17], T. Perroud *et al.* [18], Wang and Kangas [20], Loo *et al.* [23] etc. deal with locating and extracting text in color document images. These methods are by and large based on some color clustering approach (e.g. histogram-based color clustering [17,18,20], region growing [23]) followed by several other processing steps to locate textual elements in a document image. Experiments have been conducted on images of printed documents [18], of Internet pages [17, 23], scene images from digital camera [20], etc. Images are mostly well contrasted and not very much suffered

from image defects that are generally observed in historical documents.

On the other hand, studies by He and Downton [24] and Leydier *et al.* [25] are focused on separating foreground and background in color document images. The approach proposed in [24] works in HSV color space and color map in a document is identified by a color quantization algorithm that quantizes image colors into 6 levels which are found to include all distinct foreground and background colors for a batch of archive documents on which the method has been tested. The method involves a manual registration of template color maps. A foreground color is identified through its matching with the template color maps by using a fuzzy color classification algorithm.

In another technique proposed by Leydier *et al.* [25], foreground-background segmentation is achieved by a serialized k-means algorithm. However, the approach involves a manual intervention to set the number of logical classes and the color samples for each logical class to initialize the original centers of clusters in the k-means algorithm [29] which is applied on a sliding window (e.g. 6×6) so that the segmentation is neighbor-dependent. Three parameters have been used to take special measures to prevent class swapping (which is often done by an unsupervised clustering algorithm like k-means), to overcome local stains in the image, and to control the serialization of the algorithm. The algorithm has been tested with several ancient manuscript images and it observed that the parameters along with the window size which can have a heavy impact on the performance of the algorithm as well as on the computational time.

Interestingly, DjVu [27] implements an efficient foreground-background separation though in the context of compression. The approach is based on a multi-scale bicolor clustering algorithm by considering several grids of increasing resolution. Each successive grid delimits block whose size is a fraction of the size of the blocks of the previous grid. The bicolor clustering algorithm is applied on the blocks of the first grid and a foreground/background color for each block in this grid is obtained. The blocks of the next grids are then processed and this process continues until convergence of the foreground and background colors. This technique works quite well for a large category of documents in the gray as well as in color domain. However, it fails in cases where documents contain low contrasted foreground and background as observed in many handwritten manuscripts. Some demonstrations of such failure are illustrated in section 3 that presents our experimental results.

Table 1 briefly presents the major binarization and foreground extraction techniques. It no way attempts to present an exhaustive summary in this area. Rather, popular techniques are only referred, their relative strength, weaknesses and domain of application are precised.

1.2 Our approach

To design a generic document image binarization that works in both gray and color domain and can still handle a variety of degradation in documents including historical ones (pages from old printed books, handwritten manuscripts, microfilm images, etc.), we propose a new method that initially uses a connected component labeling approach to capture the spatially connected similar color pixels. This helps to rapidly locate zones containing information of interest. Next, dominant background components are determined looking at their size (in terms of member pixels) and then the entire image is the divided into number of rectangular blocks (essentially not disjoint and are of different sizes) one around each dominant background components. These blocks represent local uniformity of illumination, background, etc. and respective foreground parts are treated against these local uniformities. This provides the adaptive nature of the proposed algorithm.

Next, blocks are arranged in a forest (collection of trees) like structure as explained later in section 2.4. Blocks in a tree are hierarchically connected and parent-child relation is established based on their size and geometric layout (disjoint or overlapping). Blocks in a tree are subsequently processed in a top-down manner (i.e. from root to leaves). Components in each block undergo a bi-color clustering by traditional k-means approach where initialization is done using clustering results obtained by processing its parent block. Processing of all blocks results in binarization of an input document image.

Apart from locating the local uniformity characteristics, foreground-background separation or binarization results gains from two other aspects: (i) components represent much larger entities than isolated pixels and each component consist of spatially connected pixels having similar or near-similar color values. This aspect improves the performance of k-means because it gets less confused to cluster a component compared to the case when it deals with isolated pixels. (ii) to implement k-means, cluster centers are not initialized blindly rather the initialization is done by true background and foreground color values obtained by using some contextual information as described later in section 2.5. Such an initialization technique results in a quick convergence of k-means with better clustering output.

On completion of binarization, foreground components, in case of color image, further go through an unsupervised color clustering to detect different color regions (or elements) within the foreground region. The remainder of the paper is organized as follows. Section 2 presents the proposed binarization technique while section 3 outlines the approach for detecting individual color regions within the foreground part. Experimental results and observations are presented in section 4. Section 5

Table 1 Summary of binarization or foreground-background separation techniques

Category	Approach with references to some important works	Pros and cons
Global thresholding (as yet finds application in gray-scale domain only)	Assume histogram is bimodal; Threshold at valley point. 1979: Otsu[5], 1982: Rosenfeld[6], etc.	Simple to implement and often effective but breaks downs when illumination, background, noise characteristics are non-uniform.
Adaptive thresholding (though so far has been applied on gray scale document images but many approaches under this category can suitably be extended to include color domain)	Compute threshold for every pixel or region based on local statistics; 1985: Kittler[7], Tsai[8] 1986: Niblack[9], 1994: O'Gorman[10], 1997: Liu[11], 2000: Sauvola[4] Gatos, 2004[12], etc.	Several methods are computationally very expensive. Sometimes, the approaches used to model images defects do not hold for real life documents. Despite this, many proposed methods demonstrate their effectiveness in handling difficult situations for binarization.
Color Clustering (Has been used to extract text or foreground in color document images)	Color clustering and segmentation approach is involved. Several other processing steps are designed to implement required extraction. 1998: DjVu[27]; 2000: Kasturi[16], Lopresti[17]; 2001: Perroud[18], Wang[20]; 2002: Tsai[19]; 2003: Li[21], Nishida [22]; 2004: Loo[23], He[24], Leydier[25], Yan[26] etc.	Mostly designed for specific applications and lack in generality. Some require extensive manual interventions that makes the methods less attractive. However, application of some techniques for processing of historical documents result in very interesting and encouraging outcomes.

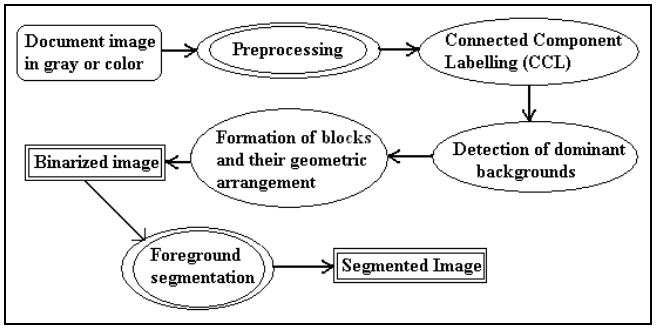


Fig. 1 Proposed approach: a schematic diagram.

concludes the paper with some discussions on future scope of research in this area.

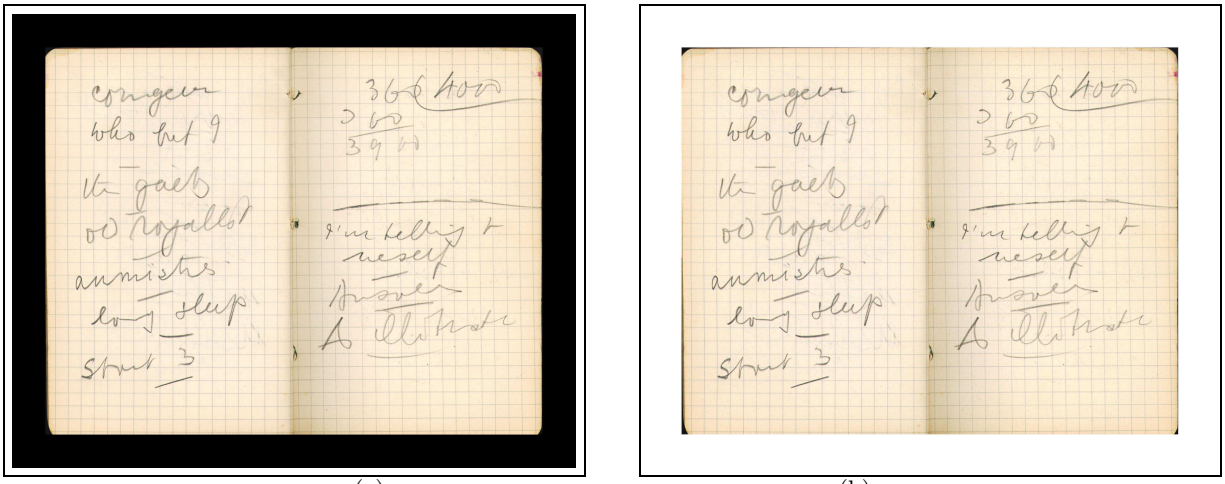
2 The proposed method

Our proposed method consists of a chain of processing steps as shown in figure 1. Optional steps are marked with double ellipses and similarly optional output requirements are marked with double rectangles. The major steps are: (i) preprocessing, (ii) connected component labeling, (iii) detection of dominant background components, (iv) segmentation of the entire image into hierarchically arranged rectangular blocks of different sizes, (v) bicolor clustering in each block and (vi) foreground segmentation. Each of these steps are explained below with some illustrations.

2.1 Preprocessing step

Preprocessing, in our approach, is treated as an optional step as it is required for certain types of images while others may directly be passed to the next stage. One of the preprocessing steps deals with color smoothing in the input image. In our application, a simple smoothing technique is involved and it is found to be efficient for a large class of documents. In this approach, each pixel color is reassigned to an average color value found within a small region surrounding that pixel. Color of each pixel (X) is redefined as the mean color computed in a 3×3 window surrounding the pixel, X (X being at the center of the window). However, for applications dealing with specific documents, one may incorporate more sophisticated smoothing techniques or image enhancement techniques like one in [13]. For example, the binarization method proposed in [12] uses a low-pass Wiener filter for this purpose.

Another preprocessing step deals with locating the paper document inside the document image. In many cases, documents mostly the historical ones contain a dark outer region surrounding the document and this often happens due to the requirement to scan such document against a dark background. Figure 2(a) shows one such document. Antonacopoulos and Karatzas [28] address this issue. They assume the real paper edges to be approximately straight and identification of outer edges starts by examining the edge pixels of the image for each of the four edges (top, bottom, left and right). From each edge pixel the process moves inwards (row or column wise) and the difference in Lightness for each pair of ad-



(a) (b)

Fig. 2 Preprocessing: identification of actual paper document.

jacent pixels is recorded. A pixel is marked as a potential paper-edge one if the difference is found to be above a threshold value or the difference in Lightness between the current pixel and the average of the previous pixels examined is above the threshold. A straight line is subsequently fitted on each of the four potential paper edges.

In our approach, we follow a simple but quite effective technique that initially computes row and column wise run-length of pixel colors. Only one color channel is used for color documents. Moreover, instead of 256 levels of a color (or gray values), a low resolution of 32 levels has been used to measure run length. Next, maximum run length is noted for each row and column and first order difference is calculated as follows. Let r_i be the maximum run length noted for the i-th row and R be the number of pixel rows. First order difference is computed as, $\delta_i = \left| r_i - r_{i+1} \right|$ for $i = 0 \ldots (\frac{R}{2} - 1)$ and $\delta_i = \left| r_i - r_{i-1} \right|$ for $i = \frac{R}{2} \ldots (R-1)$. Actual page borders (top and bottom) are found by locating two picks (top and bottom) in the histogram of δ_i. Similar operations are carried out column wise to find left and right page borders. Figure 2(b) shows the located paper region (outside is set to white) for the image in figure 2(a). This method though insensitive to small amount of skew ($\pm 5^\circ$ as verified experimentally) needs modification in case a document suffers from large amount of skew during scanning.

2.2 Connected Component Labeling (CCL)

After preprocessing step, a connected component labeling (CCL) is executed on the entire image. But as the traditional CCL algorithm assumes the input image in binary mode and only considers spatial connectivity (e.g. 4- or 8-connectivity), we modified this algorithm so that it captures color information and spatial details at the same time. The modified algorithm is presented under Algorithm-I where `push()` and `pop()` represent the traditional push/pop operations associated with a `stack` data structure.

It is to be noted that for the same image, CCL result returned by the Algorithm-I will differ in different color spaces and even within the same color space (even for gray scale image) results differ with different values of the threshold, τ. A low threshold value generates large number of connected components (resulting in excessive over-segmentation), on the other hand, higher values of the threshold wrongly combines pixels not having enough color similarity.

Figure 3 shows variations in results produced by the Algorithm-I for one example image (a small image is intentionally chosen for better understanding). Responses in four different color spaces namely, RGB, HSV, YCbCr, and CIE L*u*v* are presented and within the same color space results with various threshold (τ) values have been shown in Figure 3. The integer number given at the bottom-right corner of each image represents the number of connected components obtained for a particular threshold value within a chosen color space.

Application of the Algorithm-I, therefore, needs to have answers for two important aspects: (i) which color space must be used to execute CCL? and (ii) how to choose the threshold value (τ) automatically? Gray scale images concern only with the second aspect.

Selection of Color Space: As far as the first query is concerned, we have experimented with four different color spaces namely, (i) RGB, (ii) HSV, (iii) YCbCr, and (iv) CIE L*u*v* (details about these color spaces can be found in [14,15]). This experiment was initially conducted on a set of 20 images. We have carried out experiments to check CCL results if the color spaces (for the same image) are forced to generate the same (or

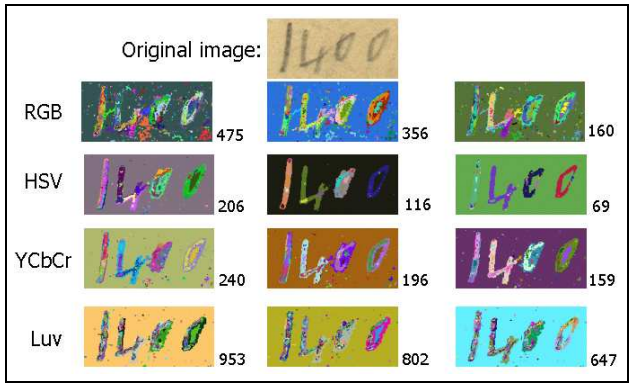


Fig. 3 Connected Component Labeling in Color Spaces: Original image is shown at the top; CCL results in a particular color space are presented row-wise; threshold value (τ) in a particular color space is gradually incremented from left to right; the number of connected components obtained for a particular threshold under a particular color space is given at the bottom-right corner of each image.

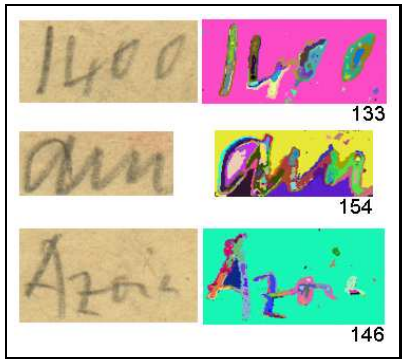


Fig. 4 Choice of threshold value for Connected Component Labeling in Color Spaces: Original images are at the left; in HSV, CCL results at an auto-selected τ are shown at the right.

nearly the same) number of components (by controlling the threshold value, (τ) in each individual color space).

It is observed that all the four spaces induce over-segmentation i.e. generate more number of components than actual ones. However, among the four spaces, a single connected component (as perceived visually) HSV is broken into less number of components and therefore, CCL under HSV generates less number of connected components for an image (images in figure 3 also well represent this phenomenon). This characteristic becomes helpful in subsequent processing stages and increases computational efficiency. Based on these observations HSV is chosen as the color space in this present experiment. Hence, choice of HSV as the color space is completely empirical rather than based on any general theory. In fact, [14] presents several merits and demerits of different color spaces to show no one is in general superior to another.

However, one may improve the CCL results by executing it in more than one color space and then producing the final results by comparing the results obtained in multiple color spaces. Otherwise, instead of using one color space at a time, more than one color space can be used to represent color of an image pixel increasing the feature dimensions used to compute μ_L and $\mathcal{D}_k$ in CCL. This has been tried by some studies like [25] where a 6-dimensional vector (RGB and HSV values together) is used to represent color feature of a pixel.

Automatic selection of threshold, τ: Considering the varying nature of documents, in our approach, we prefer to select the threshold value (τ) by some automatic means rather setting it to a predefined value. The threshold is calculated from the input image by considering only the adjacent pair of pixels in row and column wise manner. The maximum color distance between such pixel pairs are recorded for each row and column and an average of these values serves as the threshold, τ used in the Algorithm-I. A few examples of CCL results obtained at the automatically selected τ are shown in figure 4.

Algorithm-I: Connected Component Labeling (CCL) in Color Space

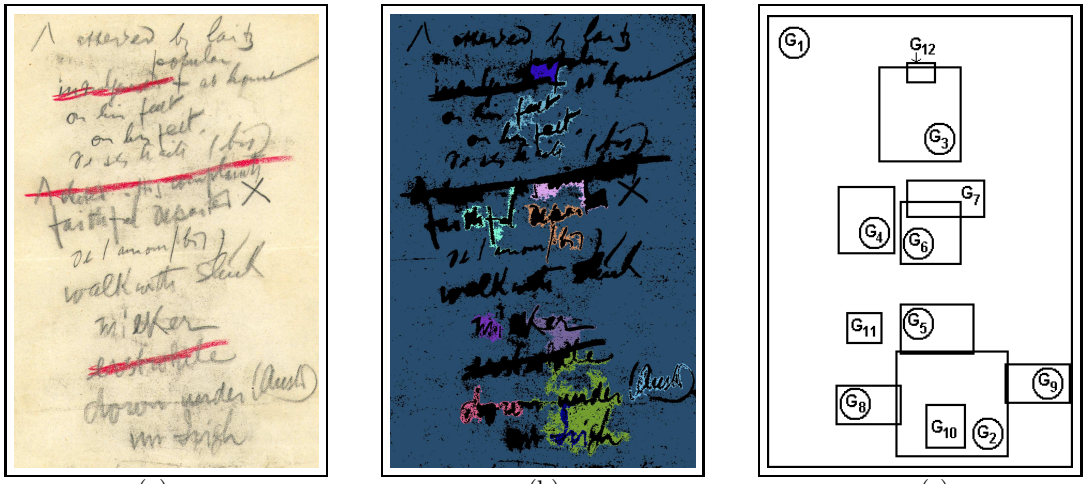
Input: Color Image (I) and Output: Labeled image consists of z number of connected components, $c_1, c_2, \ldots, c_z$.
Initialization: All pixels are initially unlabelled.
S: is the stack of pixel coordinates;
L: is the label value and initialized to 1;
for i=1 to H (image height in pixels)
  for j = 1 to W (image width in pixels)
    if X = I(i,j) is unlabelled
      push(X);
      while Stack (S) is not empty
        Y = pop();
        Label(Y) = L;
        Compute the current mean color (μ_L) for the component c_L
        If $P_1, P_2, \ldots, P_8$ are the 8-connected pixels of Y then for each unlabelled P_k
        Compute distance $\mathcal{D}_k = ||\,\text{color}(P_k) - \mu_L\,||$;
        If $\mathcal{D}_k < \tau$ (a pre-defined threshold) push(P_k);
      end while;
      L=L+1;
    end if;
  end for;
end for;

2.3 Identification of dominant background

Let $c_1, c_2, \ldots, c_z$ be the z number of connected components found after execution of the Algorithm-I. Each component (c_i is the i-th component) is represented by the following tuple:

$$< i, S_i, C_i(x,y), \text{bbox}, \mu_{h_i}, \mu_{s_i}, \mu_{v_i}, \{m_i\} > \quad (1)$$

where i is the component label, S_i is the size of c_i in terms of number of member pixels, $C_i(x,y)$ is the center



(a) (b) (c)

Fig. 5 Identification of dominant backgrounds components: (a) Input image, (b) Identified background components are painted with randomly chosen colors (black pixels are part of connected components which are either foreground or background not identified at this stage), (c) rectangular blocks formed by dominant background components.

of inertia of c_i. The term 'bbox' represents bounding box of c_i (i.e. the minimum upright rectangle surrounding c_i). Basically, 'bbox' records the top-left and bottom-right coordinates of the corresponding bounding box. The next three values in (1), namely μ_{h_i}, μ_{s_i}, and μ_{v_i} represent the mean hue, saturation and value for c_i. The last entry i.e. $\{m_i\}$ is the list of member pixels belonging to the i-th component. In fact, this list is a collection of x and y coordinates of each m_i. Obviously, this list will have S_i number of (x, y) entries one for each member pixel.

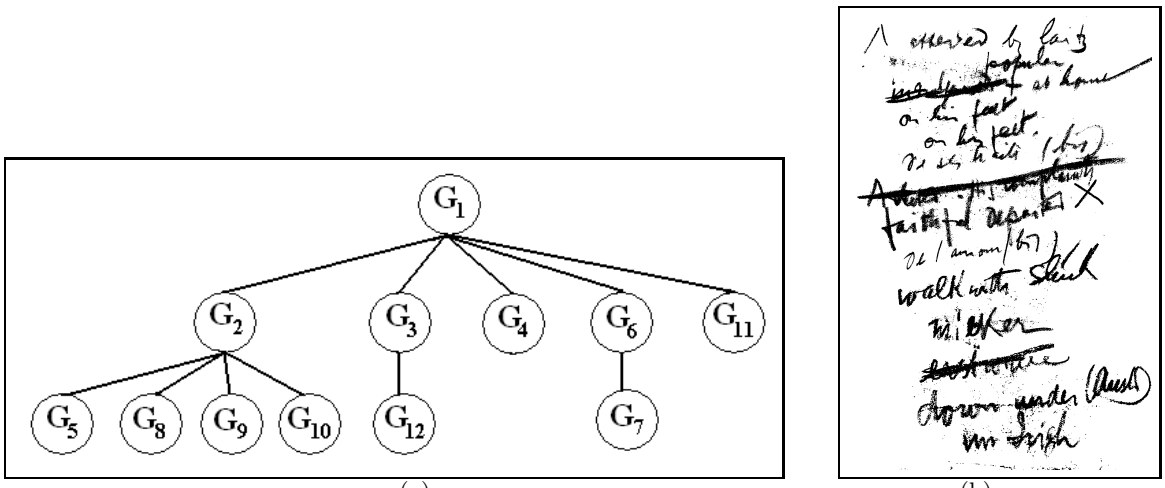
Next, the components are sorted based on their S_i (i.e. size) values and *mean* of S_i values is computed. Let μ_S be this mean. Some of background components (i.e. components representing backgrounds) are located at this stage by comparing their size against μ_S. The idea is to capture heavier or bigger (with respect to μ_S) components which are assumed to represent background parts. This assumption is quite general because the foreground parts, at large, contains only about 5% (sometimes even less) of the entire image (this is very much true for printed pages as well for handwritten manuscripts).

Therefore, if the components are compared with respect to their mean size values, it would be clear that very large (with respect to the mean size of the components) components basically represent multiple backgrounds parts which are either spatially disconnected or dissimilar in color. If the background of an image is uniform then only a few number of background components are located but this number increases if (color) non-uniformity of the background increases. The background components identified by this way captures the local uniformity of color or illumination. A foreground part, therefore, has to be identified with respect to its nearest background component instead of any global background reference. This process provides the required adaptivity of the algorithm as demonstrated later.

Initially, a component c_i is labeled as a background component if $S_i > \alpha\mu_S$, where α is a scalar whose value is not directly a user-defined one. Rather, to minimize the heuristic nature of our proposed algorithm, value of 'α' is determined dynamically. It is determined in a way so that the combined value of $\alpha\mu_S$ equals to certain percentage (in our experiment, it is set to 10%) of image area. Therefore, the value of 'α' is calculated locally and depends on the particular input image's height, width and the mean size of the components found in that image.

The components successfully pass through the above inequality condition represents dominant backgrounds in the image. Figure 5(b) shows the dominant background components extracted for the image in figure 5(a). Execution of Algorithm-I for this image results in 16,439 connected components which shows a mean size of 182.69 pixels and only 12 components have been identified as background components. Identification of these background components shows the local non-uniformity in the background of the input image. Components representing foregrounds are much smaller in sizes; however, there are various background components which are small in size and remain unidentified at this stage.

Let l be the number of identified dominant background components and $\mathcal{B}$ represents this subset of l components. Let c_b (where $b \in [1 \ldots l]$) be the biggest component and its color is taken as the reference color for background parts. Let B_{ref} denote this reference background color. A reference foreground color (F_{ref}) is then searched from the rest of all other components as follows. For each component c_i, $(i \neq b)$ its distance in color space is measured with respect to c_b and the component (c_f)



(a) (b)

Fig. 6 Foreground extraction: (a) hierarchical arrangement of blocks shown in figure 5(c), (b) extraction results after processing of blocks.

showing the highest distance is treated as reference foreground part. In other word, $||\text{color}(c_f) - \text{color}(c_b)|| > ||\text{color}(c_i) - \text{color}(c_b)|| > \forall i \neq (b \text{ or } f)$. The color of c_f is assigned as F_{ref}. Use of B_{ref} and F_{ref} will be clear in subsequent sections.

2.4 Formation of rectangular blocks and their hierarchical arrangement

After identifying the set $\mathcal{B}$, the entire image is divided into several rectangular blocks. This is done by using the bounding box information (i.e. bbox information in (1) available with each component). Hence, l blocks are identified for l background components. For example, figure 5(c) shows the blocks formed by 12 dominant background components (shown in figure5(b)) for the image in figure 5(a). Next, these blocks are arranged in a hierarchical structure following a graph theoretic approach. Geometric layouts of the blocks and the area covered by each block are considered for such an arrangement which induces a tree structure. The *parent-child* relationship between two blocks in the tree are established following the Algorithm-II.

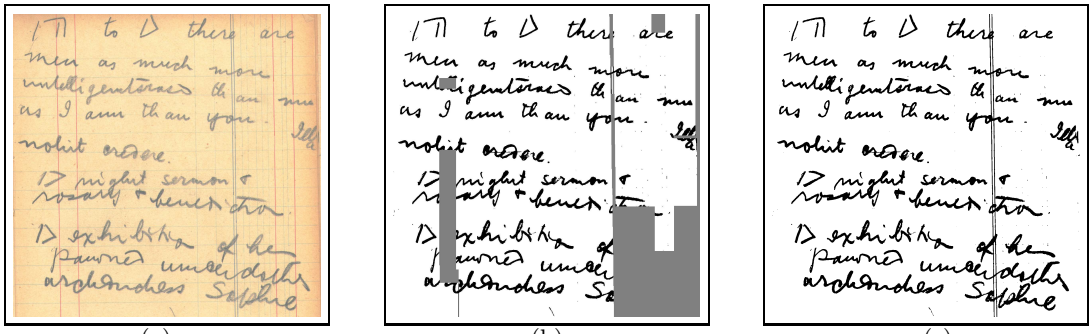
The largest block consequently forms the root of the tree and other blocks overlapping with (or contained in) the root block become the leaves and non-leaf nodes of the tree. From the Algorithm-II, it is to be noted that if a block (G_i) is overlapped with (or contained in) more than one block then the parent of G_i becomes the one that is larger than G_i but the smallest among the blocks with which G_i is overlapping (or contained in). Figure 6 demonstrates the tree that is formed with the blocks in figure 5(c). In this case, the largest block contains all the other blocks and therefore, only one tree is formed. But there may be cases where the largest block does not overlap or contain other blocks and disjoint sets of overlapping (contained in) blocks may result in. In such cases, each set of blocks generate a tree and final arrangement of blocks results in a forest (collection of trees) structure. However, in any case, the Algorithm-II defines a *parent-child* relationship, if exists between two blocks.

Algorithm-II: Hierarchical arrangement of blocks

Let $G_1, G_2, \ldots, G_l$ be the list of l-blocks corresponding to l background components.
Sort the blocks in a descending order on the area covered by them.
for i=1 to l-1
  for j = i+1 to l
    if G_i and G_j are overlapping or contained in one another
      then `parent`$[G_j]$ = G_i;
    end if;
  end for;
end for;

2.5 Bicolor clustering

The blocks as described in the preceding section are arranged in a hierarchical structure to implement a multiscale bicolor clustering of the connected components obtained after execution of CCL. In our approach, such a clustering has been achieved by traditional k-means algorithm [29] with slight modification for the initialization of the algorithm. For a tree of blocks, bicolor clustering algorithm is initially applied on root block for which one cluster (assumed to represent background) is initialized with B_{ref} and another cluster is initialized with F_{ref} obtained at the stage described under section 2.3.



(a) (b) (c)

Fig. 7 Foreground extraction: (a) Input image, (b) processed (black and white) and unprocessed (gray) regions, (c) final extraction result.

For execution of bicolor clustering algorithm for the inner blocks (leaves and non-leaf nodes) of the tree, one cluster is initialized with the same color as that of the background component represented by the block (note that each node representing a block is originated by a background component identified at the stage described in section 2.3) but the cluster representing foreground components is initialized by the foreground color found for the parent of the current node (i.e. block). This modification in initializing the bicolor clustering algorithm introduces a bias in the process to rapidly attract the components towards the true foreground and background clusters.

It is to be noted that the bicolor clustering algorithm inside a block is implemented to cluster the connected components contained by the block. The constituent components for a block is determined by looking at the $C_i(x, y)$ value (see (1)) of the components. Execution of bicolor clustering in each block classify the block's member components into background or foreground components. In fact, all components are tagged with a one-bit binary flag (this is done by adding one more binary field with attributes shown in (1)) to indicate whether a component belongs to background or foreground part. Therefore, processing of all blocks (arranged in a tree or forest) results in a binarized version of the input image. Figure 6(b) shows this end result for the image in figure 5(a).

On some occasions, the blocks representing dominant background components may not cover the entire surface of the input image. In these cases, some connected components will remain unprocessed by the bicolor clustering executed on the tree (or forest) of blocks. Therefore, a final check is done to detect unclassified connected components. This is achieved by another round of execution of bicolor clustering involving the unprocessed components only and in this case, initialization of the algorithm is done by using reference background (B_{ref}) and foreground (F_{ref}) colors obtained previously (as explained in section 2.3). Figure 7 demonstrates this process. Figure 7(a) shows the input image, figure 7(b) presents the intermediate foreground extraction result where some regions (painted in gray color) remained unprocessed as they were not covered by the blocks identified as dominant background. Final run of bicolor clustering algorithm produces the ultimate binarized version as shown in figure 7(c).

2.6 Foreground segmentation

The processes described in the preceding sections extract foreground parts from an input image. This can, in general, be viewed as binarization of the input document. However, on some occasions documents (mostly the color documents) contain foreground parts in different colors and therefore, separation of these element will probably help subsequent document analysis tasks. This section describes this process of segmenting foreground elements into different clusters each one representing foreground parts of similar color.

Let $c_1, c_2, \ldots, c_q$ be the q (out of z) connected components labeled as foreground components. Each of these q components are tagged with their mean color values as shown in (1). These components are subject to a color clustering to achieve foreground segmentation. Here the clustering is essentially an unsupervised one to avoid user interaction at this stage. The traditional *Maximin* clustering algorithm is slightly modified for this purpose. The following algorithm is a modification of the *Maximin* clustering algorithm to achieve the said task:

Algorithm-III: Clustering of Foreground components

Let $c_1, c_2, \ldots, c_q$ be the list of q-foreground components.

1. Compute $d_\tau = \dfrac{\sum_{i=1}^{q-1}\sum_{j=i+1}^{q}\mathcal{D}(c_i,c_j)}{\frac{q(q-1)}{2}}$
2. Let (c_a, c_b) where $a \neq b$ and $1 \leq a, b \leq q$ be the pair for which $\mathcal{D}(c_a, c_b)$ is maximum among all (c_i, c_j) pairs where $i = 1 \ldots (q-1)$; and $j = (i+1) \ldots q$.
3. If $\mathcal{D}(c_a, c_b) > d_\tau$ then choose $z_1 = c_a$ and $z_2 = c_b$ (cluster centers 1 and 2) and

number of clusters, $N_c = 2$.

else No segmentation is possible resulting in only one foreground region;

Termination of the algorithm.

4. Compute distance ($\mathcal{D}$) between all other samples (c_i) to $z_1, \ldots, z_{N_c}$.

Compute $d_i = \text{Min}\,(\mathcal{D}(c_i, z_j))$ for $i = 1 \ldots m$ and $j = 1 \ldots N_c$.

Compute $\text{Max}\,\{d_i\} = d_k$ for $1 \leq i \leq m$ (m denotes the no. of un-clustered components at this stage).

5. If $d_k > d_\tau$ then $N_c = N_c + 1$; $z_{N_c} = c_k$; go to step 4.

else assign remaining patterns to the closest cluster centers.

Unlike *Maximin* clustering algorithm which always finds a second cluster, our approach described in Algorithm-III seeks whether a second cluster exists (step 3 of the algorithm) using a threshold (d_τ) computed dynamically (in step 1) from the image in hand. Some of the foreground segmentation results are shown in the next section presenting experimental details.

2.7 Analysis of the Algorithm

Time and space complexity of the proposed algorithm can be analyzed in the following manner. Let $n \times m$ be the size of an input image in pixels and loading of this image needs a space of an order of $\Theta(n^2)$ (considering $n \times m \approx n^2$). Execution of the pre-processing step (section 2.1) does not demand any viable extra space but involves a time complexity of $\Theta(n^2)$. Execution of connected component labeling (CCL) of section 2.2 imposes a time complexity of $\Theta(n^2)$. Storing of the labeled image needs an extra space of $\Theta(n^2)$. However, one can avoid the use of this extra space by maintaining a label tag with each pixel in the original image itself. Moreover, implementation of CCL needs an extra space for stacking of pixels and this adds a linear space complexity of $\Theta(n)$.

Let z be number of components obtained by execution of CCL. It is to be noted that $z \ll n^2$. Storing of these z-components requires a space of $\Theta(z)$ and sorting (based on their size value) of components involves a time complexity of $\Theta(z \log z)$. This sorting is required to identify dominant background components as described in section 2.3. Let l background components are identified (induces a time complexity of $\Theta(l)$) generating l number of blocks to be arranged in a *tree* (or *forest*) structure(section 2.4). Such a data structure needs an extra space of $\Theta(l)$ and arrangement of l blocks into a *tree* (or *forest*) requires a time complexity of $\Theta(l^2)$ (see Algorithm-II). Note that complexity of Algorithm-II can be reduced to $\Theta(l \log l)$ but considering the minor gain in overall complexity, this modification was not explored in our current experiment.

Next, the bicolor clustering is executed separately on l blocks. Let a block, G_i contain o_i connected components and t_i be the number of iterations needed for the convergence of k-means in block G_i. Note that $\sum_{i=1}^{l} o_i = z$. Hence, processing (section 2.5) of each block G_i involves a time complexity of $p_i\Theta(o_i)$. Time complexity to process all blocks is therefore, $\sum_{i=1}^{l} [p_i\Theta(o_i)]$ which roughly equals to $p \times \Theta(z)$ where $p = \sum_{i=1}^{l} p_i$ and $\Theta(o_i) \approx \Theta(z)$ (though $\forall i,\ o_i \leq z$). Additional space complexity to implement bicolor clustering is quite low as just two cluster centers are maintained for each block resulting in a low constant (linear) space requirement.

Finally, the algorithm for foreground segmentation (section 2.6) induces a time complexity of $\Theta(q^2)$ for the first two steps (Algorithm-III). The rest of the steps in this algorithm are executed in $\Theta(q)$ time. Moreover, the algorithm is single-pass in nature. Like k-means the Algorithm-III also does not impose much space complexity as only space (additional) requirement is to keep information about N_c clusters found among the q foreground components. Table 2 summarizes the time and space complexity involved by the different steps of the proposed algorithm. The method shows a total complexity expressed as, $3\Theta(n^2) + \Theta(z \log z) + \Theta(l) + \Theta(l^2) + p\,[\Theta(z)] + \Theta(q^2) + \Theta(q)$ which is of the order of $\Theta(n^2)$ which comes as the dominating term in the preceding expression. Similarly, space complexity is also $\Theta(n^2)$.

3 Test data, experimental results and discussions

As the real challenge lies in dealing with low quality documents showing different degradations due to uneven illumination, aging, scanning, etc., creation of test data puts emphasis on processing of documents of this sort including historical documents mostly handwritten manuscripts of famous personalities. Both printed as well as handwritten manuscripts in color and gray-scale are considered. For the current experiment, documents dominant in text (printed or handwritten) are only taken into consideration.

The dataset contains 100 documents which are broadly divided into three parts among which first two consider color documents while other deals with gray scale documents. Most of the images are scanned at 300 dpi (a few are at 150 dpi) and images are quite large in size (average size is about 3M pixels) as often encountered in reality.

Since the proposed algorithm does not contain any training (or learning is real sense) component, there was no need of using a training data in the experiment. However, the method makes use of one user-defined threshold value, i.e. the combined value of '$\alpha\mu$' in section 2.3 (please note another parameter 'τ' used in Algorithm 1 is determined dynamically). Therefore, 20 images are initially used to (i) empirically choose the right value for '$\alpha\mu$' (in this experiment, the combined value of $\alpha\mu_S$ equals to 10% of the total image area) and (ii) choose a color space (HSV in this experiment) convenient for the present purpose. In strict sense, this set of 20 images is

Table 2 Computational complexity of the proposed method

Processing steps	Space complexity	Time complexity
Reading of input image	$\Theta(n^2)$	$\Theta(n^2)$
Pre-processing/smoothing	$\Theta(n^2)$	$\Theta(n^2)$
Connected component labeling	$\Theta(n^2) + \Theta(n)$	$\Theta(n^2)$
Storing and sorting of connected components	$\Theta(z)$	$\Theta(z \log z)$
Identification of dominant backgrounds	nil	$\Theta(l)$
Arrangement of blocks	$\Theta(l)$	$\Theta(l^2)$
Processing of blocks	$\Theta(l)$	$p \times \Theta(z)$
Foreground segmentation	$\Theta(N_c)$	$\Theta(q^2) + \Theta(q)$

Image size: $n \times m \approx n^2$; no. of connected components: z ($\ll n^2$); no. of components identified as dominant background: l ($< z$); p is the no. of iterations required for convergence of k-means; no. of foreground components: q ($< z$); no. of foreground regions based on color similarity: N_c.

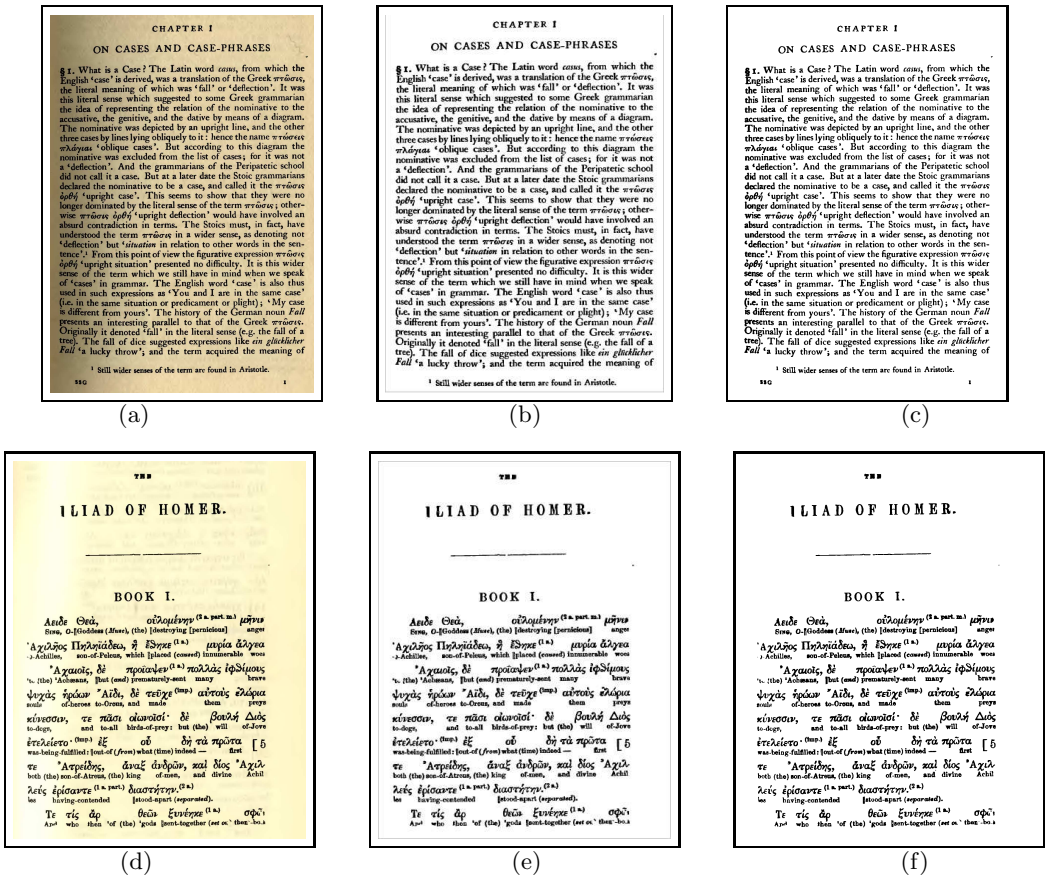


Fig. 8 Foreground extraction results on DjVu sample documents:: (a) and (d): Input images; (b) and (e): corresponding extraction results obtained (at 300 dpi) by DjVu technique; (c) and (f): results obtained the proposed approach.

not called a training set, as the algorithm has not been trained on this set. However, these images are excluded from the final test set.

The test set consisting of 80 images is divided into three parts. Under Part-I of the test set, 20 test documents are taken from the '*Ancient books, and Historical Documents*' category of '*DjVu Zone Digital Library*'[1]. Selection of these documents enable us to compare our results with those by DjVu technique [27]. Most documents under this group are century old, some are more than 200 years.

[1] Freely available at http://www.djvuzone.org/djvu/antics/index.html

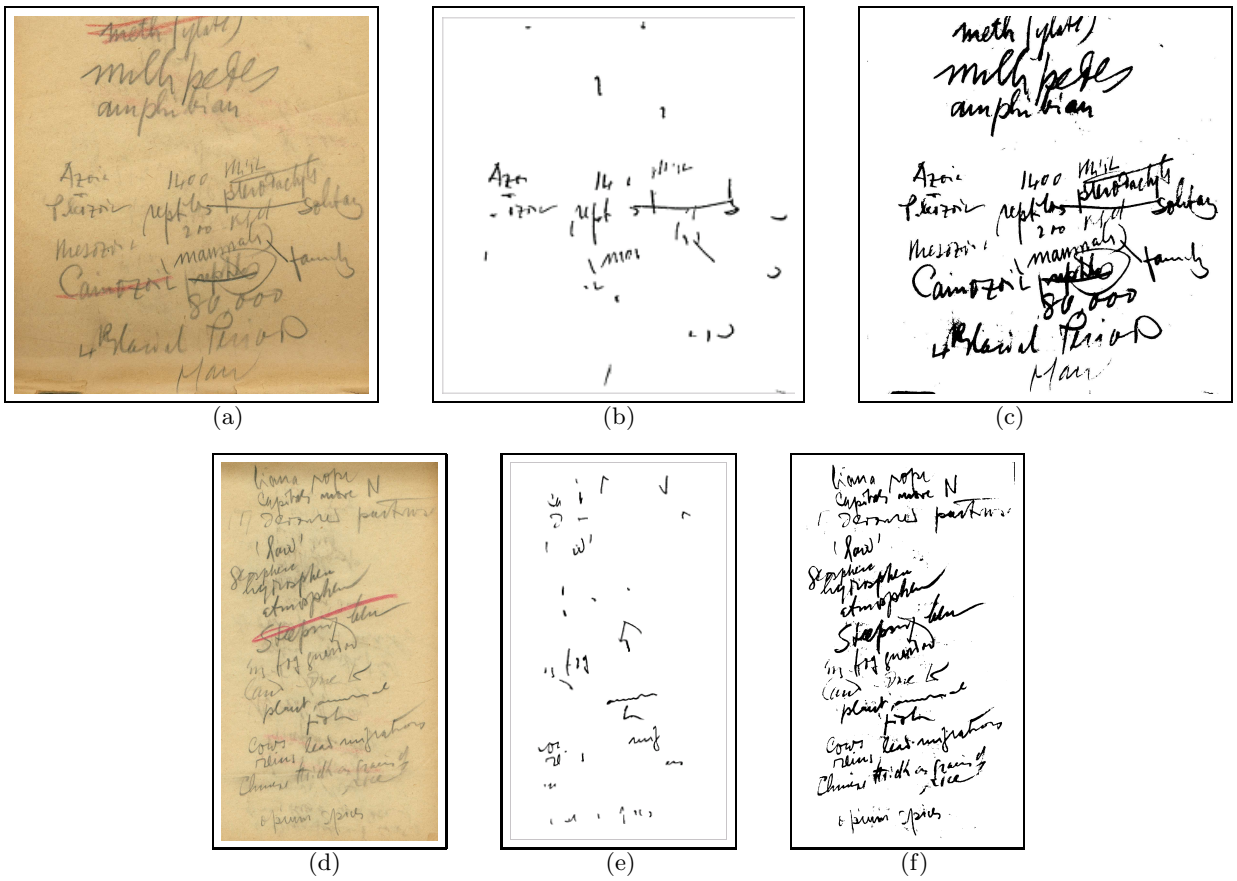


Fig. 9 Foreground extraction results for handwritten historical manuscripts:: (a) and (d): Input images; (b) and (e): corresponding extraction results obtained (at 300 dpi) by DjVu technique; (c) and (f): results obtained the proposed approach.

Part-II of the test dataset consists of 40 documents which are in color scanned from working notebooks of several famous writers of 19th and 20th century. Samples include images from the notebooks of Gustave Flaubert (1821-1880), James Joyce (1882 - 1941), etc. Many of these manuscripts had been written with quill or lead pencil (not ink) and pencil marks are frequently spread over the background. Because of the very low contrast between foreground and background parts, efficient extraction of foreground marks seems quite difficult for these documents. Part-III of the dataset considers 20 gray-scale documents which are mostly scanned images of century old handwritten manuscripts, microfilm images, etc. Out of these 20 documents, 10 are taken from manuscript of '*Madame Bovary*'[2] and others are taken from the database of correspondences received by Emile Zola (1840-1902). These correspondeces are recorded on microfilms which has been scanned into images using Canon MS 800 Microfilm scanner [3].

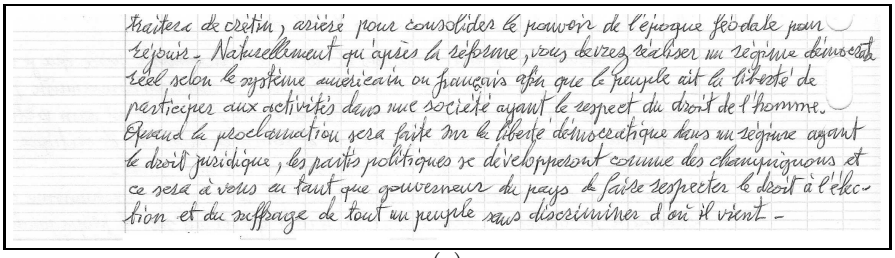
[2] Available at www.univ-rouen.fr/psi/BOVERY

[3] Available with the Bibliotheque Nationale de France, Department Des Manuscripts, N.A.F. 24.523

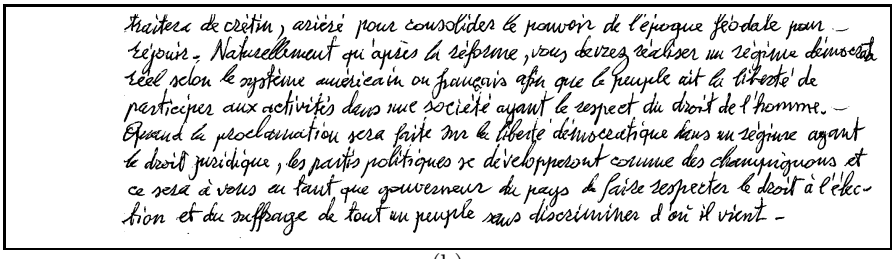
Figure 8 shows extraction results for two documents taken from Part-I of the test dataset. Figure (b) and (c) (and similarly (e) and (f)) compares the foreground extraction results obtained by DjVu [30] and our proposed technique. These documents are taken from DjVu document database where DjVu achieves good extraction results but our results are also comparable to that DjVu as checked visually. A quantitative evaluation is presented later part of this section.

Next, Part-II documents are tested. Because of the very low contrast between foreground and background parts in many of these documents, DjVu very often fails to properly extract the foreground parts, whereas our proposed method, in major cases successfully locate the foreground elements. Figure 9 presents a few examples where our technique outperforms DjVu results. Test results on Part-III documents which are in gray-scale show that the proposed approach is equally effective for gray-scale images. Some results are presented in figure 10 where (d) shows result on old handwritten manuscript scanned at gray-scale image (figure c) and (f) shows result on a microfilm image (figure e).

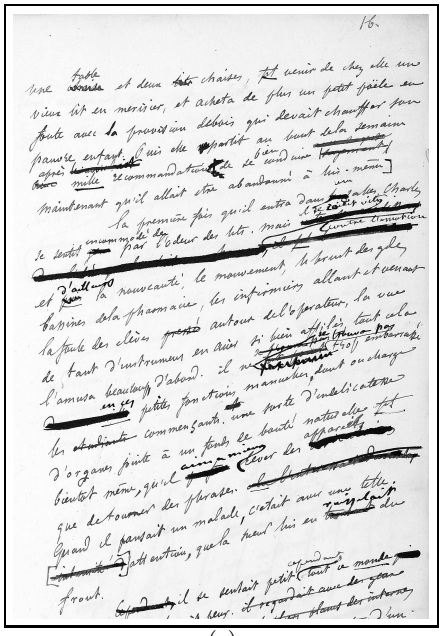
Evaluation of foreground extraction efficiency: Examination of foreground extracted from the test im-



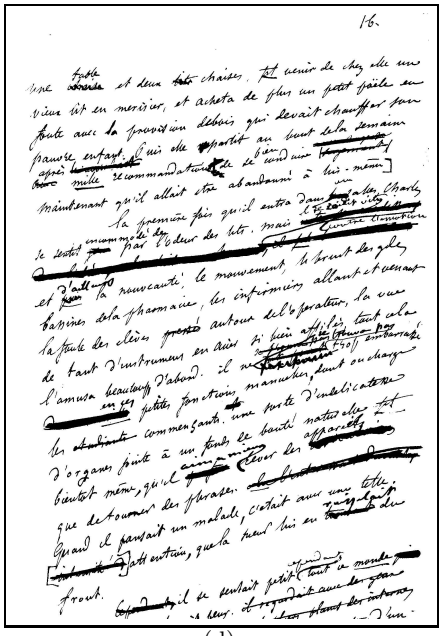
(a)



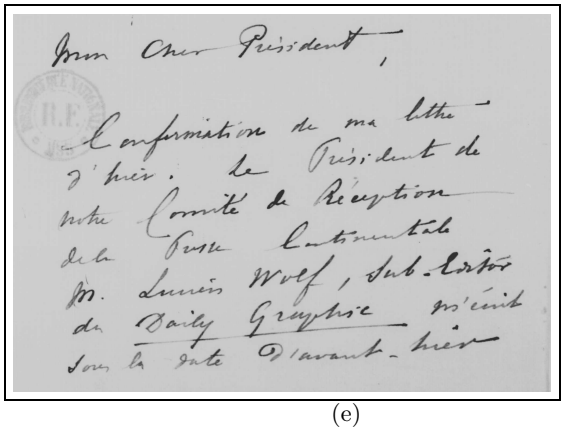
(b)



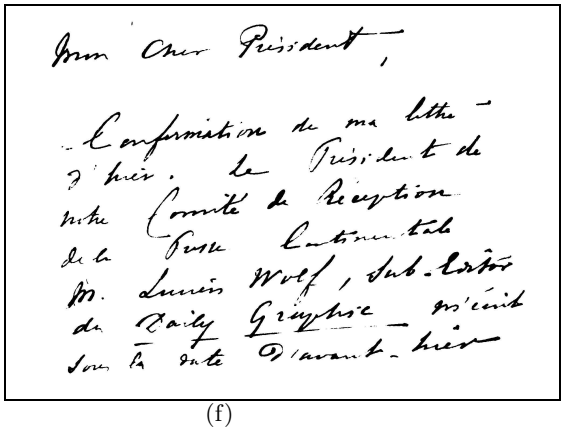
(c)



(d)



(e)



(f)

Fig. 10 Foreground extraction results for gray-scale documents:: (a), (c) and (e): Input images; (b), (d) and (f): corresponding foreground extraction results.

Table 3 Evaluation of foreground extraction results

Evaluation level	#No.	#Correctly extracted		#Partially extracted		#Missed		ϵ_l or ϵ_w	
		DjVu	Our	DjVu	Our	DjVu	Our	DjVu	Our
Line	1,457	889	1,217	431	174	137	66	0.61	0.84
Word	95,411	69,648	88,732	16,047	4,703	9,716	1,976	0.73	0.93

Table 4 Foreground extraction results: details for 10 test images

Image Sample	#Word no.	#Correctly extracted		#Partially extracted		#Missed		ϵ_w	
		DjVu	Our	DjVu	Our	DjVu	Our	DjVu	Our
1	38	1	17	2	4	35	17	0.03	0.45
2	24	0	24	1	0	23	0	0	1.0
3	23	0	21	6	2	17	0	0	0.91
4	37	14	37	11	0	12	0	0.38	1.0
5	32	1	28	4	4	27	0	0.03	0.87
6	117	116	116	1	1	0	0	1.0	1.0
7	113	113	101	0	12	0	0	1.0	0.89
8	43	43	43	0	0	0	0	1.0	1.0
9	27	27	27	0	0	0	0	1.0	1.0
10	31	2	26	6	26	23	0	0.06	0.84

ages reveals that one of the three phenomena takes place: a foreground part (visually perceived) is (i) extracted correctly, (ii) missed, or (iii) the extracted part is not a foreground part at all in the corresponding image. Therefore, an ideal method for evaluation of foreground extraction results should consider all these three aspects. However, several non-trivial problems make designing of such an ideal evaluation method extremely difficult. Basic question is what would be the unit for measuring extraction efficiency. For example, measuring efficiency at the pixel level is difficult as no groundtruth (i.e. whether a pixel belongs to foreground or background) is available for the test images.

Since the test documents are predominantly textual in nature, accuracy could have been measured at the stroke level. But designing a consistent evaluation technique working at the stroke level is equally difficult due to lack of a proper definition of stroke in the handwritten data being dealt in the present study (images shown in different figures of this correspondence well represent this fact). Therefore, quantification of accuracy of extraction is finally done at two different levels: lines and words. Evaluation of extraction results is done by manually computing the number of lines and words in each original image (I_o) and in the corresponding extracted foreground (I_e). Next, extraction efficiency (ϵ_l: at the line level, and ϵ_w: at the word level) is measured as,

$$\epsilon_l = \frac{\text{No. of lines (correctly extracted) in } I_e}{\text{No. of lines in } I_o} \quad (2)$$

$$\epsilon_w = \frac{\text{No. of words (correctly extracted) in } I_e}{\text{No. of words in } I_o} \quad (3)$$

It is to be noted that sometimes word boundaries are not very clear in some original images and therefore, the counts (mainly the word counts) are based on manual perception only. Errors in extractions are classified into two categories namely, (i) partially extracted and (ii) completely missed. A line is partially extracted if some its constituent words are missed, whereas extraction of a word is partial if some constituent strokes are missed. Table 3 presents the evaluation results for the combined set of 80 test documents. Evaluation results for DjVu technique is presented for a comparative evaluation. Table 4 presents evaluation results in details for 10 samples (handwritten manuscripts of famous writers and are considered as historical documents of special importance) that show the marked improvement achieved by our proposed method over the DjVu technique in extracting foreground parts.

Major sources behind extraction errors: Visual examination of extraction/binarization results reveals that though the proposed approach successfully works for most of the images, in a few cases, it fails to locate all the visible foreground parts of an input image. Figure 10(f) demonstrates such a problem for processing of the image (which is a microfilm image) in figure 10(e). Several strokes are broken in the extracted foreground. It is analyzed that such problems occur more in handwritten manuscripts than in printed documents and it is mainly due to (i) very weak stroke marks, (ii) very low contrast between a foreground stroke and its corresponding background and (iii) spreading of ink or pencil marks over the background.

Bleed-through (or show-through) effect which is due to seeping of ink from the reverse page side imposes another problem. It is observed that when a document having show-through effect is subject to foreground extraction, marks present due to show-through effect are sometimes identified as foreground. Figure 11 shows such an example where original image is suffered with bleed-through effect and extracted foreground contains several bleed-through strokes. However, in such cases bleed-

through marks are quite prominent and look similar (as visually perceived) to the true foreground parts and therefore, identifying them as foreground parts should not be judged as any weakness of the proposed algorithm which primarily deals with foreground extraction/binarization. Rather, some specialized technique similar to ones proposed in [13,31,32] can be used to tackle the problems related to bleed-through effect. We treat this part as an future extension of the present study.

Another problem is experienced for some kind of images where the background contains a grid like structure, some portions of the grid is also inconsistently labeled as foreground along with the actual foreground parts. Figure 12 shows such an example. Original image and extraction result are shown in figure 12(a) and (b), respectively. In this case, color similarity among the grid marks and the handwritten parts is so close that they are not separated even after foreground segmentation.

Foreground segmentation: Foreground segmentation is studied for color images to segment the foreground parts into regions based on similar colors. Performance of segmentation results is judged by checking how many color maps are properly located against the actual number of maps (marked manually) in the foreground part of an input image. Experiments show our proposed algorithm (i.e. the Algorithm-III) rarely misses any color map present in the foreground parts but on a few occasions, more than one color maps are generated for a single (as perceived visually) color map. Figure 13 shows an example of foreground segmentation where figure 13(a) shows a document with three visible color zones: (i) background (ii) handwriting text and (iii) red strokes. Figure 13(b) shows result after foreground extraction and 13(c) exhibits that two different color maps are identified within the foreground parts.

Analysis of computational efficiency: Previously in section 2.7, time and space complexity of the proposed method is discussed. The absolute run time required by the method is also analyzed. Programs (written in Matlab and C) when executed on a server (having two processors of 1.5 GHz clock speed, primary memory of 1 GB and shared by about 30 people) take on an average 8.17 seconds to produce the background/foreground separation result for images of an average size of 2000×1500 pixels. This time does not include the time for reading the image into memory. This is done by Matlab tool and it takes around 10 to 15 secs on a desktop (P-IV, 1.7 GHz) machine with 256 MB RAM. The average time units taken by intermediate steps are like these: (i) preprocessing steps: 1.25 secs; (ii) connected component labeling: 2.75 secs; (iii) arrangement and processing of blocks: 3.5 secs; (iv) foreground segmentation: 0.67 secs. This analysis further shows the computational efficiency of the technique as compared to many other existing adaptive binarization methods which are often computationally quite expensive. For example, method described in [25] takes about 500 secs to process a image of size 3000×2000 on a latest high-end PC.

4 Conclusions

In this paper, an efficient approach is presented for background/foreground separation in document images with an emphasis on processing of low quality color documents for which a few studies have been reported so far. Algorithm is tested on varieties of documents staring from gray-scale to color, printed and handwritten manuscripts, documents with well contrasted text as well as those suffering from degradations like uneven illumination, aging, etc. which quite often observed in historical documents. Test documents contain several samples scanned from handwritten manuscripts of famous writers. These manuscripts are written with quill, pencil, etc. and generate low contrast between background and foreground. Results show enormous adaptability of the proposed approach with the uneven illumination or local changes in background and foreground color.

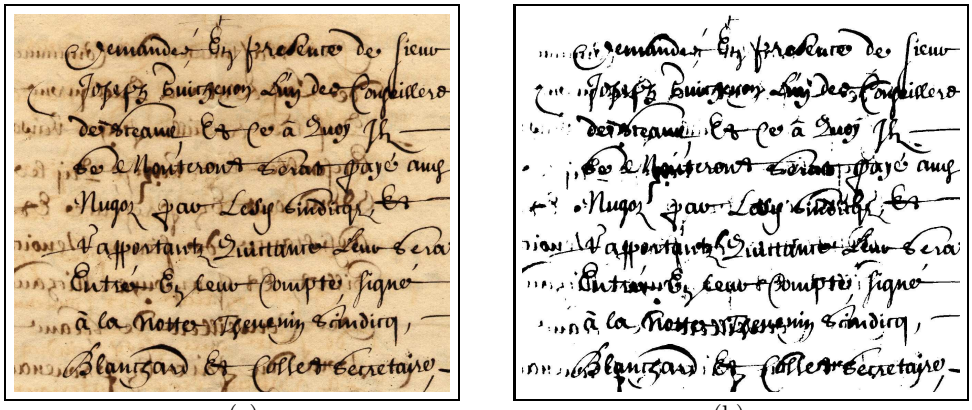
The method finds its application in the area of binarization, compression of documents where foreground and background layers are separated to achieve better compression, locating text in documents, image enhancement in digital preservation of ancient documents etc. However, the algorithm has so far been tested on text dominant documents only. Behavior of this algorithm for documents containing in non-textual elements like graphics, half-tones, etc. is considered as future extension of the present work.

Moreover, proper assessment of the extraction results needs benchmarking and groundtruthing of foreground and background pixels in sample documents. This needs use of extensive manual intervention and hence finding an efficient way (may be semi-automatic in nature) of achieving it could be treated as another future direction of the current study. Design of several other processing steps for initial enhancement of the input image, bleed-through removal, improvement in foreground segmentation results, etc. needs further research.

Acknowledgement: One of the author wishes to thank CNRS, France for providing him a postdoctoral fellowship at Laboratoire PSI - FRE CNRS 2645, UFR des Sciences, University of Rouen, France. The work presented in this paper has been carried out under this fellowship. Authors would also like to thank Prof. Frank Le Bourgeois for sharing some test images with us.

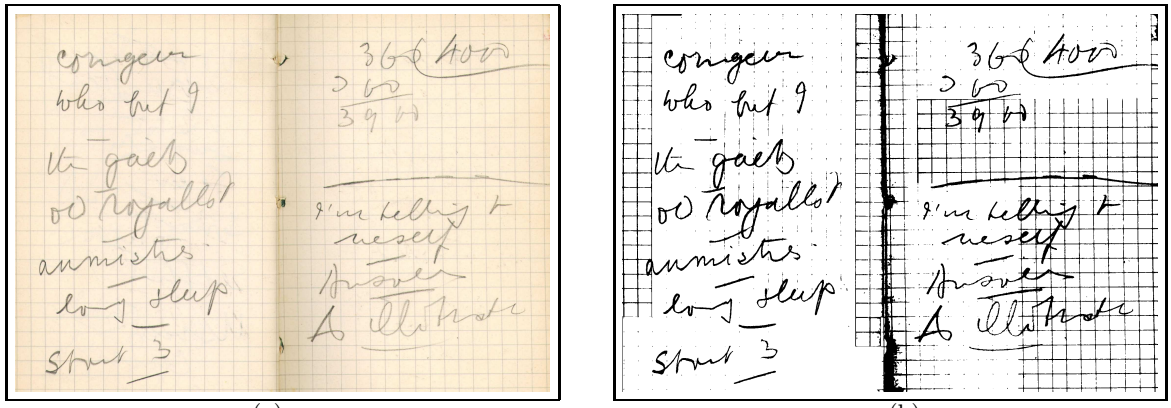
References

1. P.K. Sahoo, S. Soltani, A.K.C. Wong and Y.C. Chen, "A survey of thresholding techniques," *Comput. Vision, Graphics, Image Process.*, vol. 41, pp. 233-260, 1988.



(a) (b)

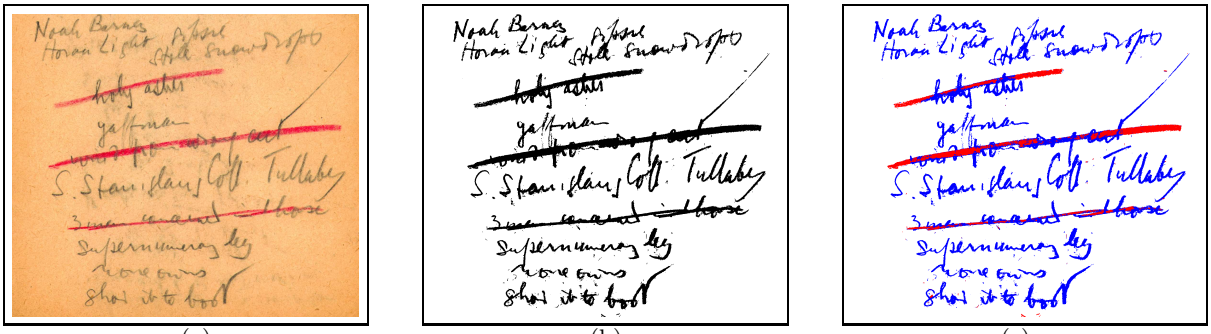
Fig. 11 Bleed-through in foreground extractions: (a) Input image; (b) extraction result.



(a) (b)

Fig. 12 Errors in foreground extractions: (a) Original image where background contains grid like structure, (b) Final result for foreground-background separation where several grid strokes are mislabeled as foreground parts.

2. O.D. Trier and T. Taxt, "Evaluation of binarization methods for document images," *IEEE Trans. Pattern Anal. Machine Intell.* vol. 17, pp. 312-315, 1995.
3. O.D. Trier and A.K. Jain, "Goal-directed evaluation of binarization methods," *IEEE Trans. Pattern Anal. Machine Intell.* vol. 17, pp. 1191-1201, 1195.
4. J. Sauvola and M. Pietikainen, "Adaptive document image binarization," *Pattern Recognition*, vol. 33, pp. 225-236, 2000.
5. N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Systems Man Cybernet.*, vol. 9, pp. 62-66, 1979.
6. A. Rosenfeld and A.C. Kak, "Digital Picture Processing," *Academic Press*, New York, 2nd edition, 1982.
7. J.Kittler and J.Illingworth, "Threshold selection based on a simple image statistic," *Computer Vision Graphics and Image Processing* CVGIP, vol. 30, pp. 125-147, 1985.
8. W.-H. Tsai, "Moment-preserving thresholding: A new approach," *CVGIP: Graphical Models Image Process.* vol. 29, pp. 377-393, 1985.
9. W. Niblack, "An Introduction to Digital Image Processing," pp. 115-116, *Prentice-Hall*, 1986.
10. L. O'Gorman, "Binarization and multi-thresholding of document images using connectivity," *CVGIP: Graphical Models Image Process.* vol. 56, pp. 494-506, 1994.
11. Y.Liu and S.N.Srihari, "Document image binarization based on texture features," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol.19, no. 5, pp.540-544, 1997.
12. B. Gatos, I. Pratikakis, and S.J. Perantonis, "An Adaptive Binarization Technique for Low Quality Historical Documents," *in 6th International Workshop on Document Analysis Systems* (DAS), Florence, Italy, September, 2004, Lecture Notes in Computer Science (LNCS), Springer-Verlag, Germany, vol. 3163 pp. 102-113.
13. Qian Wang, Tao Xia, Lida Li, Chew Lim Tan, "Document Image Enhancement Using Directional Wavelet," *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), Madison, Wisconsin, USA, June, 2003.
14. H.D. Cheng, X.H. Jiang, Y. Sun, and J. Wang, "Color image segmentation: advances and prospects," *Pattern Recognition*, vol. 34, pp. 2259-2281, 2001.
15. R.C. Gonzalez and R.E. Woods, "Digital Image Processing," *Addison-Wesley Publishing Company*, January, 2002.
16. V.Y. Mariano and R. Kasturi, "Locating uniform-colored text in video frames," *Proc. Int'l. Conf. on Pattern Recognition*, vol. 4, pp. 539-542, 2000.
17. D. Lopresti and J. Zhou, "Locating and Recognizing Text in WWW Images," *Information Retrieval,* vol. 2, pp. 177-206, 2000.
18. T. Perroud, K. Sobottka, and H. Bunke, "Text extraction from Color Documents- Clustering Approaches in Three



(a) (b) (c)

Fig. 13 Foreground segmentation:: (a) input image; (b): extracted foreground; (c): segmented foreground.

and Four Dimensions," *6th International Conference on Document Analysis and Recognition* (ICDAR), Seattle, USA, pp. 937-941, 2001.

19. C. M. Tsai and H. J. Lee, "Binarization of color document images via luminance and saturation color features," *IEEE Transactions on Image Processing*, vol. 11, No. 4, pp. 434-451, 2002.
20. K. Wang and J.A. Kangas, "Character Location in scene images from digital camera," *Pattern Recognition,* vol. 36, pp. 2287-2299, 2003.
21. Y. Li, Z. Wang, and H. Zeng, "String Extraction From Color Airline Coupon Image Using Statistical Approach," *in 7th International Conference on Document Analysis and Recognition* (ICDAR), vol. I, pp. 289-293, Edinburgh, Scotland, August, 2003.
22. H. Nishida and T. Suzuki, "A Multiscale Approach to Restoring Scanned Color Document Images with Show-Through Effects," *in 7th International Conference on Document Analysis and Recognition* (ICDAR), vol. I, pp. 584-588, Edinburgh, Scotland, August, 2003.
23. P.K. Loo and C.L. Tan, "Adaptive Region Growing Color Segmentation for Text Using Irregular Pyramid," *in 6th International Workshop on Document Analysis Systems* (DAS), Florence, Italy, September, 2004, Lecture Notes in Computer Science (LNCS), Springer-Verlag, Germany, vol. 3163 pp. 264-275.
24. J. He and A.C. Downton, "Colour Map Classification for Archive Documents," *in 6th International Workshop on Document Analysis Systems* (DAS), Florence, Italy, September, 2004, Lecture Notes in Computer Science (LNCS), Springer-Verlag, Germany, vol. 3163 pp. 241-251.
25. Y. Leydier, F.L. Bourgeois, and H. Emptoz, "Serialized k-Means for Adaptative Color Image Segmentation-Application to Document Images and Others," *in 6th International Workshop on Document Analysis Systems* (DAS), Florence, Italy, September, 2004, Lecture Notes in Computer Science (LNCS), Springer-Verlag, Germany, vol. 3163 pp. 252-263.
26. C. Yan and G. Leedham, "Decompose-Threshold Approach to Handwriting Extraction in Degraded Historical Document Images," *in 9th Int. Workshop on Frontiers in Handwriting Recognition* (IWFHR), Kokubunji, Tokyo, Japan, 2004.
27. L. Bottou, P. Haffner, and P.G. Howard, "High Quality Document Image Compression with DjVu," *Journal of Electronic Imaging*, vol 7, no 3, pp 410-425, SPIE, 1998.
28. A. Antonacopoulos and D. Karatzas, "A Complete Approach to the Conversion of Typewritten Historical Documents for Digital Archives," *in 6th International Workshop on Document Analysis Systems* (DAS), Florence, Italy, September, 2004, Lecture Notes in Computer Science (LNCS), Springer-Verlag, Germany, vol. 3163 pp. 90-101.
29. J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proc. of the 5th Berkeley Symposium on Mathematics, Statistics, and Probabilities*, vol. 1, pp. 281-297, Editors: J. Neyman and L.M. LeCam, Berkeley and Los Angeles (Calif), University of California Press, 1967.
30. Document Express DjVu Editor (Pro) 4.1.0. Copyright (c) 2000-2003, *LizardTech, Inc.*
31. E. Smigiel, A. Belaid, and H. Hamza, "Self-Organizing Maps and Ancient Documents," *in 6th International Workshop on Document Analysis Systems* (DAS), Florence, Italy, September, 2004, Lecture Notes in Computer Science (LNCS), Springer-Verlag, Germany, vol. 3163 pp. 125-134.
32. A. Tonazzini, E. Salerno, M. Mochi, and L. Bedini, "Bleed-Through Removal from Degraded documents Using a Color Decorrelation Method," *in 6th International Workshop on Document Analysis Systems* (DAS), Florence, Italy, September, 2004, Lecture Notes in Computer Science (LNCS), Springer-Verlag, Germany, vol. 3163 pp. 229-240.

Utpal Garain received both of his B.E. and M.E. in Computer Science and Engineering from Jadavpur University, Kolkata in 1994 and 1997, respectively and PhD from Indian Statistical Institute, Kolkata in 2005. Mr. Garain started his career as a software professional in industry and later on joined as a research personnel at the Indian Statistical Institute, where he is currently a full-time faculty member. He is one of the key scientists involved in the development of a bilingual (Devanagri & Bangla) OCR system, the first of its kind in India. Mr. Garain's areas of interest are in digital document processing including optical character recognition for Indian language scripts, online character recognition, document data compression, artificial immune system, etc.

Thierry Paquet Thierry PAQUET received the Ph.D. degree from the University de Rouen in 1992 in the field of Pattern Recognition. From 1992 to 2002 he has been appointed has a Senior Lecturer at the University of Rouen where he taught Signal and Image Processing. From 1992 to 1996 he was involved in an industrial collaboration with Matra MCS and the French Postal Research Center (SRTP) for the automatisation of mail sorting and bank checks reading. During this period he also worked on stochastic models and Information Criteria. Thierry PAQUET was appointed as a full professor in 2002 at the University of Rouen. His current research area concern Handwritten Document processing including Biometry, Writer adaptation of recognition systems, handwritten document categorization for industrial purposes, complex layout analysis for historical document analysis. Thierry PAQUET is the president of the French association for research in written communication (GRCE).

Laurent Heutte Laurent Heutte (30/05/1964) received his Ph.D. degree in Computer Engineering from the University of Rouen, France, in 1994. From 1996 to 2004, he was a Senior Lecturer in Computer Engineering and Automatic Control at the University of Rouen. Since 2004, he has been a Professor in the same university. Professor Heutte's present research interests are multiple classifier systems, off-line cursive handwriting analysis and recognition, handwritten document layout analysis and information extraction from handwritten documents. Since 2003, he his an Associate Editor of Pattern Recognition journal and the representative member of the French association of pattern recognition (AFRIF) in the Governing Board of the IAPR.