COCOCLUST: Contour-based Color Clustering for Robust Binarization of Colored Text

T Kasar and AG Ramakrishnan
Medical Intelligence and Language Engineering Laboratory
Indian Institute of Science, Bangalore, INDIA - 560 012
{tkasar, ramkiag}@ee.iisc.ernet.in

Abstract

This paper proposes COCOCLUST, a contour-based color clustering method which robustly segments and binarizes colored text from complex images. Rather than operating on the entire image, a 'small' representative set of color pixels is first identified using the contour information. The method involves the following steps: (i) Identification of prototype colors (ii) A one-pass algorithm to identify color clusters that serve as seeds for the refining step using k-means clustering (iii) Assignment of pixels in the original image to the nearest color cluster (iv) Identification of potential candidate text regions in individual color layer and (v) Adaptive binarization. We propose a robust binarization technique to threshold the identified text regions, taking into account the presence of inverse texts, such that the output image always has black text on a white background. Experiments on several complex images having large variations in font, size, color, orientation and script illustrate the robustness of the method.

1. Introduction

The use of digital cameras for image acquisition has enabled human interaction with any type of document in any environment. In addition to imaging hard copy documents, digital cameras are now used to acquire text information present in 3-D real world objects such as buildings, vehicles, road signs, billboards and T-shirts rendering the images more difficult for any recognition task. Such camera-captured documents are generally characterized by varying illumination, blur, perspective distortion and deformations. Moreover, large variations in font style, size, color, orientation and layout pose a big challenge to document analysis. Conventional optical character recognition engines meant for document images obtained using flat-bed scanners fail on images acquired by this promising mode. Specialized techniques are required to deal with these problems. Thus, research on camera-based document image analysis is growing [4].

In most document processing systems, a binarization process precedes the analysis and recognition procedures. It is critical to achieve robust binarization since any error introduced in this stage will affect the subsequent processing steps. The simplest and earliest method is the global thresholding technique that uses a single threshold to classify image pixels into foreground or background classes. Global thresholding techniques are generally based on histogram analysis [6, 9]. It is simple, fast and works well for scanned images that have well-separated foreground and background intensities. Camera-captured images often exhibit non-uniform brightness because it is difficult to control the imaging environment unlike the case of the scanner. The histograms of such images are generally not bi-modal and a single threshold can never yield an accurate binary image. As such, global binarization methods are not suitable for camera images.

Local binarization techniques use a dynamic threshold across the image according to the local image statistics and offer more robustness to non-uniform illumination and background noise. These approaches are generally window-based and the local threshold for a pixel is computed from the gray values of the pixels within a window centred at that particular pixel. In Niblack's method [8], the sample mean $\mu(x, y)$ and the standard deviation $\sigma(x, y)$ within a window W centred at the pixel location (x, y) are used to compute the threshold $T(x, y)$ as follows:

$$T(x, y) = \mu(x, y) - k \times \sigma(x, y), \quad (1)$$

The constant parameter k is set to 0.2. In [11], Trier and Jain evaluated the performance of 11 popular local thresholding methods on scanned documents and reported that Niblack's method performs the best for optical character recognition. However, Niblack's method produces a noisy output in smooth regions since the expected sample variance becomes the background noise variance. Sauvola and

Pietikainen [10] address this drawback by introducing a hypothesis that the gray values of the text are close to 0 (Black) while the background pixels are close to 255 (White). The threshold is computed with the dynamic range of standard deviation (R) which has the effect of amplifying the contribution of standard deviation in an adaptive manner.

$$T(x,y) = \mu(x,y)\,[1 + k\,(\frac{\sigma(x,y)}{R} - 1)] \tag{2}$$

where the parameters R and k are set to 128 and 0.5 respectively. This method overcomes the effect of background noise and is more suitable for document images. However, as pointed out by Wolf *et al.* in [13], the Sauvola's method fails for images where the assumed hypothesis is not met and accordingly, the former proposed an improved threshold estimate by taking the local contrast measure into account.

$$T(x,y) = [1-a]\mu(x,y) \; + \; aM + a\frac{\sigma(x,y)}{S_{max}}[\mu(x,y) - M] \tag{3}$$

where M is the minimum value of the grey levels of the whole image, S_{max} is the maximum value of the standard deviations of all windows of the image and a is a parameter fixed at 0.5. This method combines Savoula's robustness with respect to background textures and the segmentation quality of Niblack's method. The Wolf's method, however, requires two passes since the parameter S_{max} is obtained only after the first pass of the algorithm.

Local methods offer more robustness to the background complexity, though at a cost of higher computational complexity. The performance of these methods depend on the size of the window used to compute the image statistics. They work well if the window encloses at least 1 character. For large fonts, where the text stroke is wider than the window, undesirable voids appear within the text stroke. This puts a constraint on the maximum font size and limits their application only to known document types. In addition, all these methods require a priori knowledge of the polarity of the foreground-background intensities and hence cannot handle documents that have inverse text. Kasar *et al.* [7] address these issues by employing an edge-based approach that derives an adaptive threshold for each connected component (CC). Though it can deal with arbitrary font size and the presence of inverse text, its performance significantly degrades, like most CC-based methods do, in the presence of complex backgrounds that interfere in the accurate identification of CCs. It also uses script-specific characteristics to filter out edge components before binarization and works well only for Roman script.

Most approaches [5, 12, 14] for the analysis of color documents involve clustering on the 3D color histogram followed by identification of text regions in each color layer using some properties of text.

Badekas *et al.* [2] estimate dominant colors in the image and CCs are identified in each color plane. Text blocks are identified by CC filtering and grouping based on a set of heuristics. Each text block is applied to a Kohonen SOM neural network to output only two dominant colors. Based on the run-length histograms, the foreground and the background are identified to yield a binary image having black text in white background. The performance of these methods rely on the accuracy of color reduction and text grouping, which are not trivial tasks for a camera-captured complex document image. The method does not consider isolated characters for binarization. Zhu *et al.* [15] proposed a robust text detection method that uses a non-linear Niblack thresholding scheme. Each CC is described by a set of low level features and text components are classified using a cascade of classifiers trained with Adaboost algorithm. An accurate identification of CCs is the key to the success of these algorithms. Complex backgrounds and touching characters can significantly degrade their performance.

In this paper, we introduce a novel color clustering approach that robustly segments the foreground text from the background. Text-like regions are identified and individually binarized such that the foreground text is assigned black and the background white regardless of its color in the original input image.

2 COCOCLUST for color segmentation

We propose a novel contour-based color clustering technique that obviates the need to specify the number of colors present in the image and to initialize. Rather than operating on the entire image, a representative set of color pixels is first identified using the contour information. This significantly reduces the computational load of the algorithm since their number is much smaller than the total number of pixels in the image. A single-pass clustering is then performed on the reduced color prototypes to identify color clusters that serve as seeds for a subsequent clustering step using the k-means algorithm. CCs are accurately identified since text and background objects fall into separate color layers. Based on the assumption that every character is of a uniform color, we analyze each color layer individually and identify potential text regions for binarization. Figure 1 shows the schematic block diagram of the proposed method.

2.1 Determination of color prototypes

The segmentation process starts with color edge detection to obtain the boundaries of homogeneous color regions. Canny edge detection [3] is performed individually on R, G and B channel and the overall edge map $\mathcal{E}$ is obtained as follows:

$$\mathcal{E} = \mathcal{E}_R \cup \mathcal{E}_G \cup \mathcal{E}_B \tag{4}$$

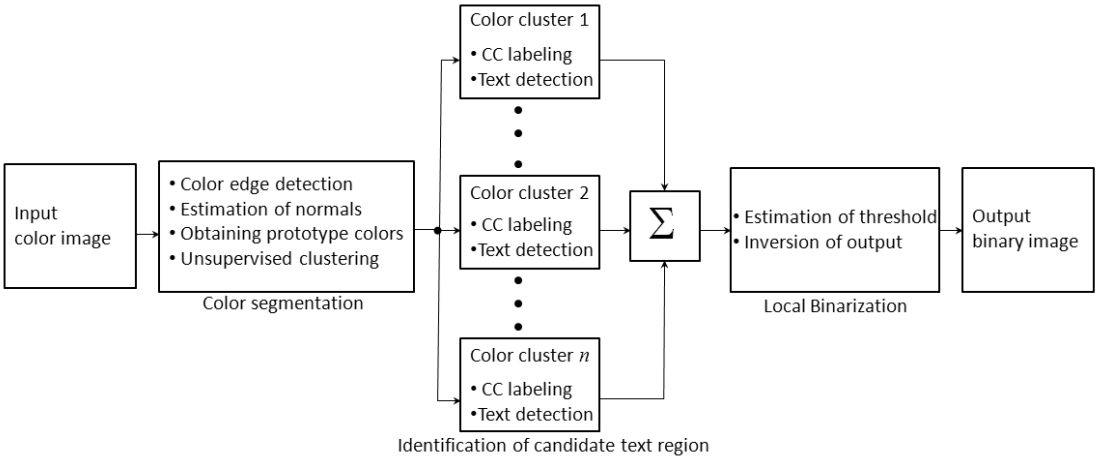


Figure 1. Block diagram of the proposed binarization method.

where $\mathcal{E}_R$, $\mathcal{E}_G$ and $\mathcal{E}_B$ are the edge images corresponding to the three color channels and $\cup$ denotes the union operation. The resulting edge image gives the boundaries of all the homogeneous color regions present in the image.

An 8-connected component labeling is performed on the edge image to obtain a set of M disjoint components

$$\{\mathcal{CC}^j\} \quad j = 1, 2, \cdots, M \text{ such that } \bigcup_{j=1}^{M} \mathcal{CC}^j = \mathcal{E}$$

The boundary pixels are identified and represented as follows:

$$\mathbf{X}_i^j = \{x_i, y_i\} \quad i = 1, 2, \cdots, n_j \tag{5}$$

where n_j is the number of pixels that constitutes the boundary and the index j refers to the connected component $\mathcal{CC}^j$. Our method employs a few vectors normal to the edge contour for every $\mathcal{CC}^j$. To estimate the normal vector, the edge contour is smoothed locally as follows:

$$\bar{\mathbf{X}}_i^j = \left\{ \frac{1}{s} \sum_{i-\frac{s-1}{2}}^{i+\frac{s-1}{2}} x_i, \; \frac{1}{s} \sum_{i-\frac{s-1}{2}}^{i+\frac{s-1}{2}} y_i \right\} \tag{6}$$

where s defines the span of pixels over which smoothing is performed and is set to 5 in this work. Here, the index i takes a circular convention to maintain continuity of the contour. The normal vectors are then computed from the smoothed contour using the following relation.

$$\mathbf{n}_i^j = \begin{bmatrix} cos(\frac{\pi}{2}) & -sin(\frac{\pi}{2}) \\ sin(\frac{\pi}{2}) & cos(\frac{\pi}{2}) \end{bmatrix} \times \frac{1}{2}\left(\frac{\bar{\mathbf{X}}_i^j - \bar{\mathbf{X}}_{i-1}^j}{\|\bar{\mathbf{X}}_i^j - \bar{\mathbf{X}}_{i-1}^j\|} + \frac{\bar{\mathbf{X}}_{i+1}^j - \bar{\mathbf{X}}_i^j}{\|\bar{\mathbf{X}}_{i+1}^j - \bar{\mathbf{X}}_i^j\|} \right) \tag{7}$$

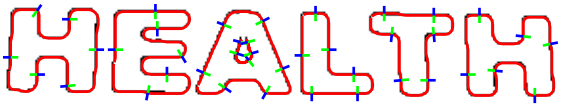
Here, the subscript i denotes the position of the boundary pixel at which the normal vector is computed and $\|\cdot\|$ denotes L_2 norm.

Since the edge image gives the boundaries of homogeneous color regions, the color values of a few pixels that lie normal to the contour are 'good' representatives of the colors present in the image. Figure 2(a) shows the edge contours for a sample color image and Figure 2(b) illustrates the selection of color prototypes from the pixels that lie normal to the edge contour. The median color values of the pixels in the normal direction that lie 'inside' ($\mathbf{n}_-^j$) and 'outside' ($\mathbf{n}_+^j$) the boundary are computed based on 3 pixels each to obtain 2 color prototypes from each normal. The color difference between two points with the same Euclidean distance in the RGB color space does not reflect the same change in the perceived color. So, we use a uniform color space, namely CIE $L^*a^*b^*$, in which similar changes in color distance also correspond to similar recognizable changes in the perceived color. The color prototypes (CP), thus obtained, are stacked column-wise as follows.

$$CP = \begin{Bmatrix} L_{1-}^* & L_{1+}^* & \cdots & L_{N-}^* & L_{N+}^* \\ a_{1-}^* & a_{1+}^* & \cdots & a_{N-}^* & a_{N+}^* \\ b_{1-}^* & b_{1+}^* & \cdots & b_{N-}^* & b_{N+}^* \end{Bmatrix}$$

where N is the number of normals along which the color values are sampled. In this work, we sample the color values from 6 regularly spaced points along the boundary of each CC yielding a total of 12 M colors. This set of color values, though much smaller in number than the total number of pixels, captures all the colors present in the image. This offers a significant advantage in terms of cheaper computation and provides an effective initialization of k-means algorithm regardless of the complexity of image content.

(a)



(b)

Figure 2. (a) A sample color image (b) Its edge contours and the computed normals that guide the selection of color prototypes. From each normal, one color value each is obtained from the pixels that lie 'inside' (Green segment) and 'outside' (Blue segment) the contour.

2.2 Unsupervised color clustering

A single-pass clustering is performed on color prototypes, as obtained above, to group them into clusters. The pseudo-code for the clustering algorithm is given below.

```
Input: Color prototypes, CP={C1, C2, ..., C2N}
       Color similarity threshold, Ts
Output: Color clusters, CL
1.  Assign CL[1]=C1 and Count =1
2.  For i = 2 to 2N, do
3.    For j = 1 to Count, do
4.      If Dist(CL[j],Ci) ≤ Ts
5.        CL[j] = Update Mean(CL[j])
6.        Next i
7.      Else
8.        Count = Count + 1
9.        CL[Count] = Ci
10.   EndIf
11.   EndFor
12. EndFor
```

where Dist(C_1, C_2) denotes the distance between the colors $C_1 = (L_1^*, a_1^*, b_1^*)^T$ and $C_2 = (L_2^*, a_2^*, b_2^*)^T$ and is computed as follows:

$$\text{Dist}(C_1, C_2) = \sqrt{(L_1^* - L_2^*)^2 + (a_1^* - a_2^*)^2 + (b_1^* - b_2^*)^2} \quad (8)$$

The threshold parameter T_s decides the similarity between two colors and hence the number of clusters. Antonacopoulos and Karatzas [1] perform grouping of color pixels based on the criterion that only colors that cannot be differentiated by humans should be grouped together. The threshold below which two colors are considered similar was experimentally determined and set to 20. We use a slightly higher threshold to account for the small color variations that may appear within the text strokes. In our implementation, the threshold parameter T_s is empirically fixed at 45, after trial and error.

The color clusters, thus obtained, are used as seeds for a subsequent clustering step using the k-means algorithm. Each resulting cluster is then examined for 'compactness' by computing distances of all the pixels in that cluster from its mean color. The maximum intra-cluster distance from its mean color is ensured to be less than 75 % of T_s if required by recursively splitting non-compact clusters into two using k-means algorithm initialized with the mean color and the one that is furthest from it. The color clusters obtained at the end of this splitting process are then used as the initial seed colors for a final pass of k-means clustering. Note that the whole clustering process is performed only on the selected prototypes. Finally, each pixel in the original image is assigned to the closest color cluster.

3 Adaptive binarization

Each color layer is then individually analyzed and text-like components are identified. We filter out the obvious non-text elements by making some sensible assumptions about the document. The aspect ratio is constrained to lie between 0.1 and 10 to remove highly elongated components. Components larger than 0.6 times the image dimensions are removed. Furthermore, small and spurious components with areas less than 8 pixels are not considered for further processing.

Text components have well-defined boundaries and hence have a high degree of overlap with the edge image as compared to non-text components. The boundary $\mathbf{X}^j$ of a component $\mathcal{CC}^j$ and its corresponding edge image region $\mathcal{E}^j$ are first dilated and their intersection is computed as a measure of stability of the boundary (BS).

$$BS^j = \frac{Area((\mathcal{E}^j \oplus S_3) \cap (\mathbf{X}^j \oplus S_3))}{Area(\mathbf{X}^j \oplus S_3)} \quad (9)$$

where S_3 is a 3×3 structuring element. Components that yield a BS measure of more than 0.5 are selected for binarization from all the color layers. Overlapping CCs are merged and fed to the binarization module to generate the desired output.

3.1 Estimation of threshold

The binarization technique proposed in [7] automatically computes the threshold value from the image data without

the need for any user-defined parameter. We use a similar approach to binarize each CC by estimating its foreground and background intensities. The foreground intensity of a component $\mathcal{CC}^j$ is computed as the mean gray level of its boundary pixels.

$$\mathcal{FG}^j = \frac{1}{n_j} \sum_{(x,y)\in \mathbf{X}^j} \mathcal{I}(x,y) \tag{10}$$

where $\mathcal{I}(x,y)$ denotes the intensity value at the pixel position (x,y) and n_j is the number of pixels that constitute the boundary $\mathbf{X}^j$.

Rather than using the bounding box to obtain an estimate of the background intensity [7], we use the available contour information that yields a more reliable decision for inversion in the presence of inverse text. Bounding boxes can have a significant overlap for inclined text and touching text lines that can result in incorrect inversion of the binary output. The contour is traced in a clock-wise direction and the normals are estimated. The background intensity is then computed as the median intensity value of the pixels along the normal direction 'outside' the boundary of the CC.

$$\mathcal{BG}^j = \text{Median}(\mathcal{I}(x,y)) \quad (x,y) \in \mathbf{n}_+^j \tag{11}$$

Note that the boundary of the CC is always 'closed' unlike during the prototype color identification stage where we may have broken as well as bifurcating edge contours. The CC is binarized using the estimated foreground intensity as the threshold value.

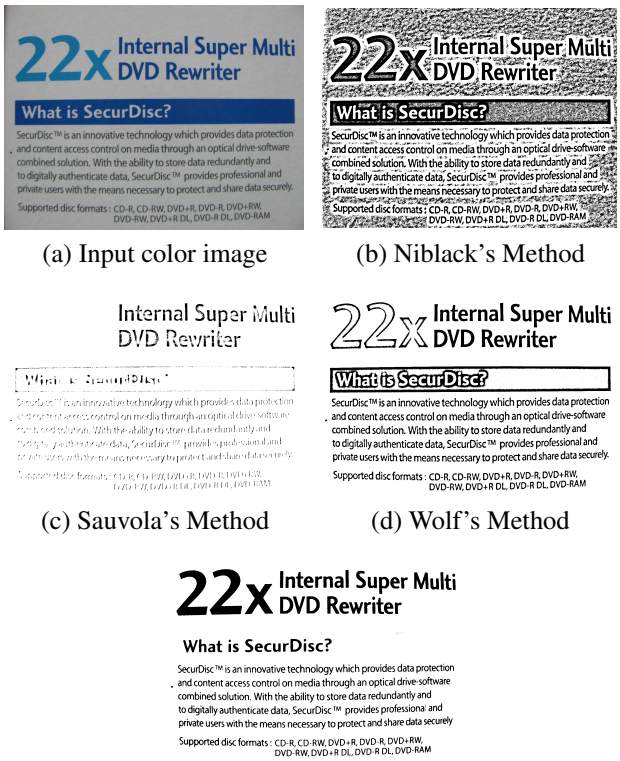
$$\mathcal{O}_j(x,y) = \begin{cases} 1 & \text{if } \mathcal{CC}^j(x,y) \geq \mathcal{FG}^j \\ 0 & \text{if } \mathcal{CC}^j(x,y) < \mathcal{FG}^j \end{cases} \tag{12}$$

The estimated values of the foreground and background intensities indicate their relative polarity. Whenever the estimated foreground intensity is higher that that of the background, the binary output is inverted to ensure that text is always represented by black pixels.

4 Experiments and results

The test images used in our experiments include physical documents such as books and charts as well as non-paper documents like text on 3-D real world objects. These images are characterized by complex backgrounds, irregular text orientation and layout, overlapping text, variable fonts, size, color, multiple scripts and presence of inverse text.

Figure 3 compares the results of our method with some popular local binarization techniques, namely, Niblack's method, Sauvola's method and Wolf's method on a document image having multi-colored text and large variations in sizes with the smallest and the largest components being 4×3 to 174×245 respectively. Clearly, these local binarization methods fail when the size of the window is smaller



(a) Input color image (b) Niblack's Method

(c) Sauvola's Method (d) Wolf's Method

(e) Proposed method

Figure 3. Comparison of some popular local binarization methods for a document image having multiple text color and size. While the proposed method is able to handle characters of any size, all other methods fail to binarize properly the components larger than the size of the window and require a priori knowledge of the polarity of foreground-background intensities as well.

than stroke width. The size of the window used here is 33 × 33. While small text regions are properly binarized, large characters are broken up into several components and undesirable voids occur within the character stroke. It requires a priori knowledge of the polarity of foreground-background intensities as well. On the other hand, our method automatically derives the threshold from the image without any user-defined parameter. It can deal with characters of any font size and color.

Figure 4 shows the result of the proposed method on images having inverse text, multiple scripts, background objects touching the text and cursive letters. The method proposed in [7] is sensitive to the background and several instances of text get filtered out since it uses an edge-based segmentation. Moreover, it uses script-dependent characteristics and works well only for isolated Roman letters.

Figure 4. (i - iv): Input images having graphic objects, inverse text, multiple scripts, complex backgrounds and cursive letters. (v - viii): The corresponding binary outputs obtained using the method proposed in [7]. Clearly, the method is sensitive to background objects since it relies on the edge information to locate CCs. It also invokes script-dependent characteristics and works well only for isolated Roman letters. (ix - xii): Binarized output images obtained with the proposed method. Color segmentation provides robustness to background complexity as well as independence to script.

In Figure 4(v) and (vii), the background and graphic objects touch some text regions and they get filtered out. In Figure 4(vi-viii), the script-dependency of the method is clearly observed. In addition to the merged characters and cursive text being eliminated, Figure 4(viii) also shows an instance of incorrect inversion of the binary output due to overlapping text lines. In contrast, the new method shows a marked improvement thanks to the color clustering algorithm that enables an accurate identification of CCs. The color decomposition effectively disambiguates background objects interfering with text as they are separated into different color layers. Each CC is individually binarized based on a threshold derived from its foreground and background intensity estimates. The background intensity estimate obtained using the contour normals provides a reliable decision to invert the binary output in the the presence of inverse text. As desired, all the text components are represented by black on white background regardless of their colors in the original image. The method is tested on several images and is found to have good adaptability. More results of the proposed method on various input images that have inverse text, arbitrary text orientation and layout are shown in Figure 5.

5 Conclusions

This paper describes an important preprocessing step for the analysis of color document images. The use of the contour information makes the method robust to the complexity of the input image. This is a desirable feature for processing camera-based images that are generally characterized by arbitrary content and layout. It does not require a priori knowledge of the number of colors present or their initialization. The contour information is successfully exploited both in color segmentation that enables accurate identification of CCs and in the inversion of the binary output to deal with inverse text. Preliminary results on camera-captured images with variable fonts, size, color, orientation, script

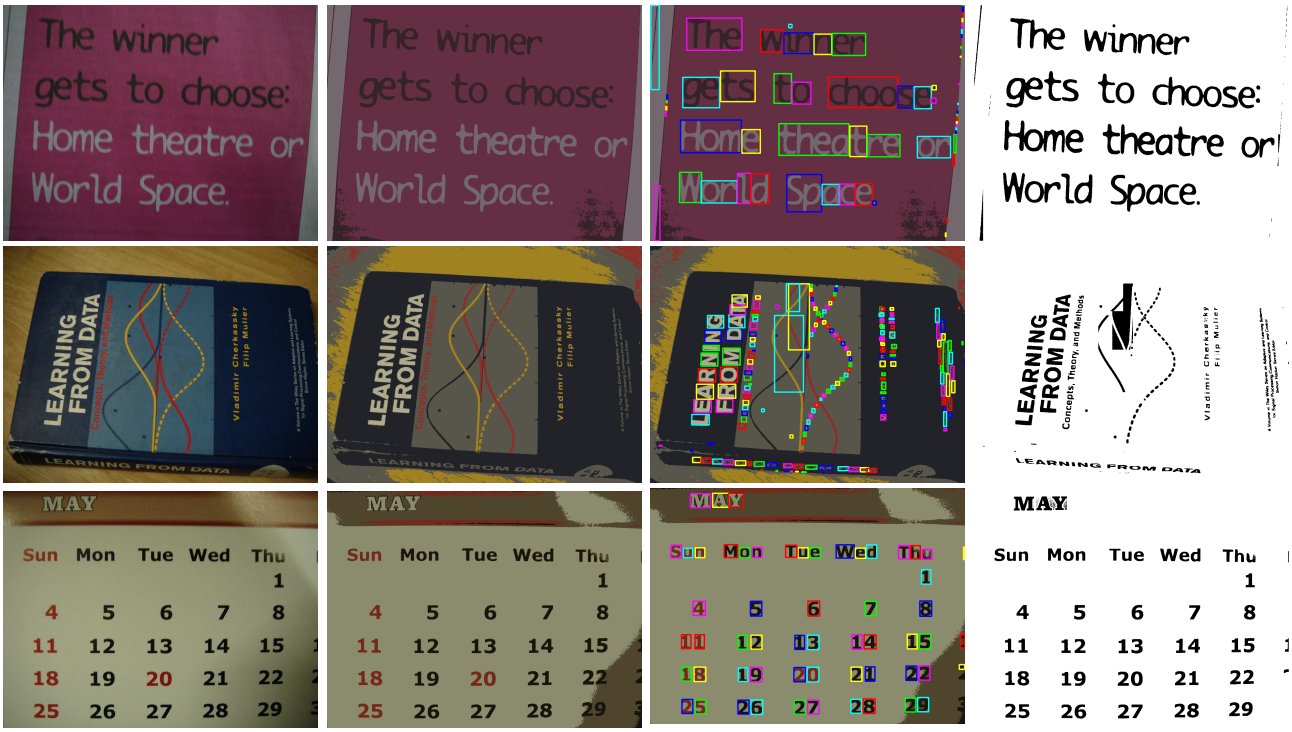


Figure 5. Input images and the corresponding color clusters, identified text regions and binarized outputs shown column-wise.

and the presence of inverse text are encouraging.

Our future work is to augment the method with a trained classifier for robust extraction of only the text regions.

References

[1] A. Antonacopoulos and D. Karatzas. Fuzzy segmentation of characters in web images based on human colour perception. *Proc. Workshop Doc. Anal. Systems*, pages 295–306, 2002.

[2] E. Badekas, N. Nikolaou, and N. Papamarkos. Text binarization in color documents. *Intl. Jl. Imaging. Syst. Technol.*, 16:262–274, 2007.

[3] J. Canny. A computational approach to edge detection. *IEEE Trans. PAMI*, 8(6):679–698, 1986.

[4] D. Doermann, J. Liang, and H. Li. Progress in camera-based document image analysis. *Proc. Intl. Conf. Doc. Analy. Recog.*, 1:606–616, 2003.

[5] A. Jain and B. Yu. Automatic text location in images and video frames. *Pattern Recog.*, 3(12):2055–2076, 1998.

[6] J. N. Kapur, P. K. Sahoo, and A. Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Comp. Vision Graphics Image Process.*, 29:273–285, 1985.

[7] T. Kasar, J. Kumar, and A. G. Ramakrishnan. Font and background color independent text binarization. *Proc. Intl. Workshop Camera Based Doc. Anal. Recog.*, pages 3–9, 2007.

[8] W. Niblack. *An Introduction to Digital image processing*. 1986.

[9] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. Systems Man Cybernetics*, 9(1):62–66, 1979.

[10] J. Sauvola and M. Pietikainen. Adaptive document image binarization. *Pattern Recog.*, 33:225–236, 2000.

[11] O. D. Trier and A. Jain. Goal-directed evaluation of binarization methods. *IEEE Trans. PAMI*, 17(12):1191–1201, 1995.

[12] B. Wang, X. F. Li, F. Liu, and F. Q. Hu. Color text image binarization based on binary texture analysis. *Proc. IEEE Intl. Conf. Acoustics, Speech Sig. Proc.*, 3:585 – 588, 2004.

[13] C. Wolf, J. Jolion, and F. Chassaing. Text localization, enhancement and binarization in multimedia documents. *Intl. Conf. Pattern Recog.*, 4:1037–1040, 2002.

[14] Y. Zhong, K. Karu, and A. Jain. Locating text in complex color images. *Pattern Recog.*, 28(10):1523 – 1535, 1995.

[15] K. Zhu, F. Qi, R.Jiang, L. Xu, M. Kimachi, Y. Wu, and T. Aizawa. Using adaboost to detect and segment characters from natural scenes. *Proc. Intl. Workshop Camera Based Doc. Anal. Recog.*, pages 52–59, 2005.