Text Localization, Extraction and Inpainting in Color Images

Mohammad Khodadadi*, and Alireza Behrad**
* M.Sc. Student, Faculty of Engineering, Shahed University Tehran, Iran, khodadadi@shahed.ac.ir
** Assist. Prof., Faculty of Engineering, Shahed University Tehran, Iran, behrad@shahed.ac.ir

Abstract: *in this paper, we propose a new algorithm for subtitle text detection, extraction and text inpainting in color images. The proposed algorithm includes three stages. In the first stage, we localize text blocks to extract subtitles. We then extract the text characters precisely and in the third stage, we use an inpainting algorithm to recover the original texture of the image. To localize text blocks, we use stroke filter and new segmentation and verification algorithms based on the image profiles. To extract text character precisely, we estimate background and text color in the candidate blocks using the color histogram measured in different sub-blocks of the candidate text blocks. Finally, the inpainting algorithm based on matching algorithm is utilized to reconstruct the initial image contents in text areas. Our inpainting algorithm considers priority for inpainting of pixels, which results in perfect reconstruction in areas with texture variation. We tested the proposed algorithm with different image and video frames and compared the results with those of other methods. Experimental results showed the efficiency of the proposed algorithm.*

Keywords: Text detection, Text localization, Inpainting.

1. Introduction

Nowadays, text detection and recognition in image and video frames is a widespread research area. Text subtitles in video, and images are used to give more information about the image or video. Text detection and recognition are used in different applications like video and image retrieval. It is also necessary in some applications to extract the added texts in the image and recover the original image. For this purpose, it is required to extract texts and inpaint the original image. Inpainting algorithms usually aim to reconstruct the original image for deteriorated images or videos.

Text detection in an image or video frame contains two stages. In the first stage, the locations of text blocks are determined and in the second stage, the precise areas of text characters in blocks are determined.

Several approaches have been proposed for text localization in images. Existing approaches for text localization can be roughly divided into three groups, including 1- color or intensity based approaches [1, 2], 2- gradient based method [3] and 3- methods based on text classifier [4-7].

Color based approaches assume a predefined color for the text. Therefore, they lose their generality when the text color is not determined. These methods analyze connected component after color segmentation for the text localization [2].

Gradient based approach utilizes the high contrast of text areas and the directional intensity variation for the text area detection. These approaches generally employ the directional gradient of the image for text localization. Gradient based approaches can be utilized both in color and intensity images.

Text classifiers are based on the structural or statistical features of the image blocks. The features are utilized to train a classifier for text area detection. These methods are mostly used for the verification of the extracted text blocks and are rarely used directly for text area localization.

Different algorithms have been proposed for the text and character extraction. Document binarization is the mostly used method for text extraction. Local thresholding algorithms like methods proposed by Niblack [8], Wolf et al. [9], and Sauvola and Pietikainen [10] estimate different thresholds for image pixels. Some algorithms like Sauvola and Pietikainen [10] method may also combine local and global thresholding algorithms.

After extracting text characters, the inpainting algorithm starts, which aims to reconstruct the original image in the text areas.

Several approaches for image inpainting have been introduced, which may be classified into three groups:

1- Algorithms based on image interpolation, which mostly utilize Partial Differential Equation (PDE). [11-15].
2- Texture synthesis based on image structure [16-19].
3- Combination of interpolation and texture based approaches [20-25].

The first group yield good results in thin damaged regions. However, in thick regions, they result in blurry edges. They also suffer from high computational cost. The BSCB algorithm [9], Total Variational (TV) [12,15] and Curvature-Driven Diffusion (CDD) [11] are based on PDE algorithm, which take color information contents around the damaged regions for the interpolation and iterate the algorithm to reconstruct the damaged area.

In algorithms based on texture synthesis, a region around the damaged pixel is searched to find a texture similar to the texture of the damaged pixel. Then the

978-1-4673-1148-9/12/$31.00 ©2012 IEEE

damaged pixel is replaced with a pixel with similar texture [12, 13].

The third group is the combination of two previous approaches. Although this method has high computational burden, it works well in filling damaged regions. The method decomposes the input image to different textures and structures and applies texture synthesis and PDE methods respectively.

In this paper, we propose a new algorithm for text localization, extraction and inpainting. The proposed algorithm for text localization is based on the image gradient; however an efficient segmentation and verification algorithm are utilized to enhance the efficiency of the proposed algorithm. To extract text character, we estimate background and text colors in the candidate text blocks using the image histogram. Then based on the text color histogram texts areas are determined precisely.

We have used fast inpainting algorithm to fill character areas, which is based on texture synthesis. This algorithm is the combination of erosion operation, matching, and new idea of repairing damaged pixels with priority.

The paper is organized as follows. Section 2 represents text localization algorithm. Proposed algorithm for extracting characters precisely is described in section 3. Section 4 represents inpainting algorithm for text area filling. Experimental results appear in section 5 and we conclude the paper in section 6.

2. Text localization

The aim of the text localization algorithm is to detect text blocks in the image or video frames. To detect text block, we first apply the stroke filter as it is defined in Equation 1 [26, 27]. This filter is a kind of derivative operator which produce better results in character strokes than Canny or Sobel operator.

$$I_s(p) = \max_{v \in N(p)} \{I(v)\} - \min_{v \in N(p)} \{I(v)\} \quad (1)$$

where $I_s(p)$ is the output of stroke filter, $I(v)$ is the intensity of color RGB channels and $N(p)$ denotes the pixels in the neighborhood of pixel p. Stroke filter are applied to different channels of input channels.

After applying the stroke filter, we keep the points with high values and set the remaining pixel to zero. For this purpose, we first calculate an adaptive threshold value using Xiaojun Li method [19]. The method combines local and global information of the stroke image to calculate threshold values as follows:

$$T_l = \max(m_l + \alpha_l \sigma_l, m_g + \alpha_g \sigma_g) \quad (2)$$

Where m_g and σ_g are global mean and variance of stroke image and m_l and σ_l are local mean and variance measured in 18*18 windows. To remove points with weak stroke filter output, we use the following equation:

$$I_{sb}(p) = \begin{cases} 1 & I_s(p) > T_l \\ 0 & o.w. \end{cases} \quad (3)$$

$$I_{st}(p) = \begin{cases} I_s(p) & \sum_{v \in N(p)} I_s(v) > T_n \\ 0 & o.w. \end{cases} \quad (4)$$

where $I_{st}(p)$ is thresholded stroke image and $N(p)$ denotes the pixels in the neighborhood of pixel p.

The thresholded stroke filter highlights edge area in the image, however text areas generates double edges in the image. To highlight text areas, we use density filter to fill holes between text edges. The density image is calculated using the following equation [28]:

$$I_{dens}(x,y) = \frac{1}{(2w+1)(2h+1)} \sum_{m=-w}^{w} \sum_{n=-h}^{h} I_{st}(x+m, y+n) \quad (5)$$

We use 10*6 windows to calculate density image $I_{dens}(x,y)$. We then apply a threshold to density image to extract text areas. The threshold value is calculated using the following equation:

$$TD = 30 + k\left[\frac{1}{N}\sum_y \sum_x I_{dens}(x,y) - 30\right] \quad (6)$$

where k is set to 0.35. Fig. 1 shows the results of stroke and density filters applied to the R channel of input image. We also applied dilation operator to enhance the output.

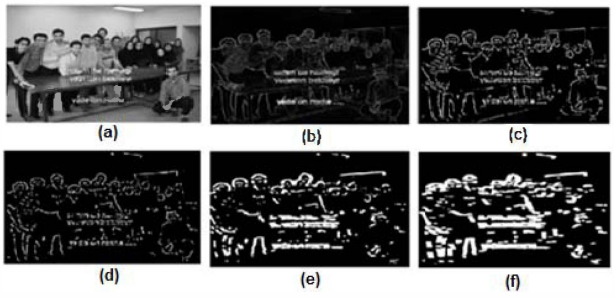

Fig. 1: The results of stroke and density filters applied to the *R* channel of input image. (a) *R* channel of the input image, (b) the output of stroke filter, (c) thresholded stroke image, (d) density image, (e) thresholded density image, and (f) the result after applying dilation.

In the next stage of the algorithm, we divide the thresholded density image to separated text blocks. For this purpose, we scan the thresholded density image using 100*200 windows image with the overlap of 50*100. Then the horizontal and vertical projections of binary pixels are used to divide the block into smaller blocks respectively. We detect local minimums in the projection profiles to divide blocks. We then select sub-blocks with proper contents and repeat the split process again to obtain proper blocks. Fig. 2 shows the process of extracting blocks.

After extracting initial text blocks, we calculate the density and area of the blocks to select proper text blocks. For this purpose, we keep only blocks with the following conditions:

Fig. 2: Dividing thresholded density image to blocks, (a) dividing using horizontal projection profile, (b) dividing using vertical projection profile, (c) the second stage of splitting using horizontal projection profile (d) the second stage of splitting using vertical projection profile.

- Blocks with the density of more than 0.75.
- Blocks with the length and height of more than 8 pixels.
- Blocks with the area of more than 200 pixels.

We then merge blocks with overlap or close blocks. In this case, the bounding box of merged blocks is considered as the new block. Merging the blocks may create inappropriate blocks again; therefore we use projective profiles to correct the bounding box again. We then remove blocks with small length or height as well as small area.

The blocks obtained until this stage, may also contain non-text areas. Therefore, we utilize verification stage to find and remove non-text areas. To verify text area, we utilize again vertical and horizontal projection profiles of binary points in the blocks. We apply a threshold to projection profiles using the mean and standard deviation of profiles as follows:

$$T_{ph} = m_{ph} + 0.5 * \sigma_{ph} \quad (7)$$

$$T_{pv} = m_{pv} + 0.5 * \sigma_{pv} \quad (8)$$

where m_{ph} and m_{pv} are the mean values of horizontal and vertical profile, σ_{ph} and σ_{pv} are the standard deviations and T_{ph} and T_{pv} are threshold values for the horizontal and vertical profiles in each merged blocks respectively. Then the number of profile elements which their value is higher than the threshold value is calculated. A block is considered as text block if the following conditions are satisfied.

$$N_{ph} > 0.2h \quad (9)$$

$$N_{pv} > 0.4w \quad (10)$$

$$N_{pv} < 0.95w \quad (11)$$

where h and w are the block width and height respectively and N_{ph} and N_{pv} are the number of elements in horizontal and vertical profiles, which are greater than their corresponding threshold values.

Furthermore, we check the block for the enough number of edge points in the block. For this purpose, we utilize the thresholded stroke image of Equation 4 and calculate the edge point's number in the block. Fig. 3 shows different stages of the block extraction algorithm.

3. Text Extraction

The aim of the text extraction algorithm is to extract text area from the candidate blocks. We suppose that the text in each block has the same color; however the background may be complex. The aim of the proposed algorithm is to inpaint the text area; therefore the precise extraction of the text area is mandatory for our algorithm.

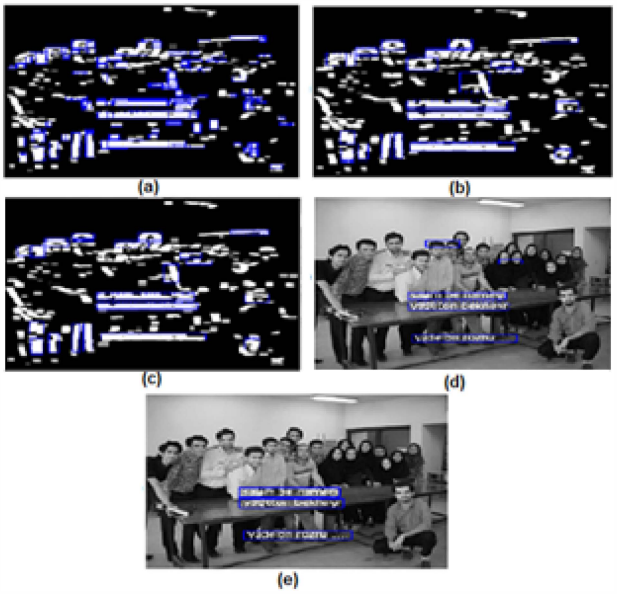

Fig. 3: Text block extraction algorithm, (a) initial text blocks (b) text blocks after merging the overlapped and close blocks (c) text blocks after correcting merged bounding boxes (d) text blocks after the verification using the projection profiles (e) results after applying edge point's number constraint.

For this purpose, we estimate histogram of color channels for the text and background areas in order to extract the text characters in the candidate blocks.

The proposed text extraction algorithm starts by the estimation of the histograms of color channels for text and background areas. We calculate the histogram for each channel individually. To estimate histogram for background, we consider two areas with the height of two pixels above and below the candidate block as it is shown in Fig 4. Then the histogram of image pixels in these areas is calculated. Fig. 5 shows the estimated histogram for R channel of background area of a sample candidate block. As it is shown in this figure, we smooth the histogram curve to remove noisy peaks.

To estimate histogram of color channels for text areas, we first extract pixels in the boundary of text characters. For this purpose, we employ thresholded stroke image of Equation 4 and determine pixels in the candidate block with high intensity change. Then these pixels are utilized to estimate the histogram for color channels. The pixels in the boundary may belong to background area as well. Therefore, we compare the obtained histogram with the histogram of background and remove peaks that are close to the peaks in the histogram of background. This algorithm works well in text areas with medium or high contrast. However, the algorithm may fail when the text and background have very similar colors.

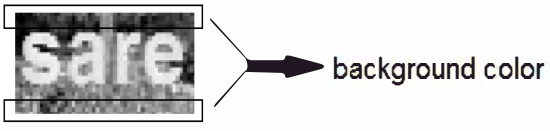

Fig. 4: The areas for the estimation of background histogram.

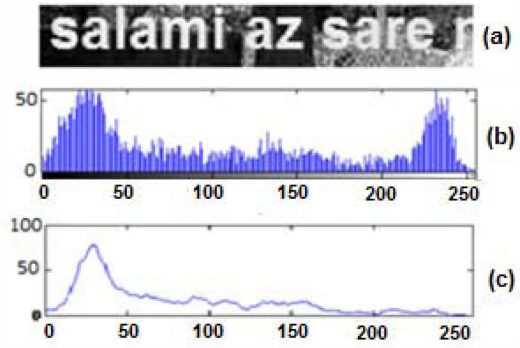

Fig. 5: The histogram of color channels for background area (a) candidate text block, (b) histogram of R channel, (c) smoothed histogram after applying 1-D averaging filter.

To handle this problem, we have devised an algorithm which can extract text areas efficiently in low contrast text blocks as well. The algorithm includes the following stages:

- Estimate and smooth the histogram of color channels as stated before.
- By utilizing the thresholded stroke image determine pixels in the boundaries of text areas.
- Divide the candidate text blocks to non-overlapping sub-blocks with the height of the block and the width of 50 pixels.
- In each sub-block calculate the histogram of text area using the pixels obtained in step 2.
- Smooth the histogram of color channels for the text area and determine the peaks.
- Compare the peaks in the histogram of text area with those of in the background histogram and remove peaks in the histogram of text area, which are close to the peaks in the background histogram.
- Determine the peak with maximum value in the histogram of text area in each sub-block.
- Sort the selected peak values and select the peak with median value as the text color peak.
- Apply a threshold to the text color peak and determine the text colors. We experimentally use half of the peak value as the threshold value.

As mentioned before, we process each color channel individually. When the text areas are determined by each channel individually, their outputs are simply combined by logical OR operation. We finally apply some post processing to fill holes and determine final text areas.

4. Text Inpainting

The proposed inpainting algorithm is based on texture synthesis and matching. The algorithm finds best matching pixel for the damaged pixels and repairs them based on the matched pixels. The algorithm includes the following stages:

- Select proper damaged pixel for the inpainting. The proper damaged pixel is a pixel which has the smallest number of damaged pixels in its 8-neighbors.
- Search in 8*8 windows centered on the selected damaged pixel for the best matching pixel. The best matching pixel is a pixel, which is more similar to damaged pixel. To measure similarity, we have used Sum Squared Difference (SSD) criterion as follows:

$$r(d_x,d_y)=\sum_{c=R,G,B}\sum_{x=-w/2}^{x=+w/2}\sum_{y=-w/2}^{y=+w/2}(I_c(x,y)-I_c(x+d_x,y+d_y))^2 \qquad (12)$$

where $r(d_x,d_y)$ represent similarity value, w is the window size to measure similarity and c is the window's color to measure the similarity.

- Replace the color of the selected damaged pixel with the color of matched pixel.

The sequence or the priority of selecting damaged pixel for reconstruction has a major effect on the efficiency of the inpainting using texture matching algorithm. Fig. 6-(a) shows the results of standard texture matching algorithm for the inpainting of image with two different textures. As it is shown in the figure, the algorithm cannot reconstruct the border completely.

To handle this problem, we have employed an algorithm, which utilizes the new idea of repairing damaged pixels with priority. The algorithm includes the following stages:

- Find the borders of the damaged region.
- Inpaint the pixels in the borders using matching algorithm and priority based on less damaged pixels in the 8-neighbors.
- Find the pixels in the border and detect the pixels with high intensity change.
- Inpaint pixels with the high intensity variation.
- Repeat the inpainting stages to fully reconstruct the damaged region.

To detect pixels with high variation, we calculate the intensity variation using 3*3 window and SSD approach. Then we apply a threshold to determine pixels with high variation. The threshold value is calculated based on the maximum value of intensity variation.

Fig. 6-(b) shows different stage of inpainting using the proposed algorithm. As it is shown in the Fig. 6-(b), the proposed algorithm reconstructs the damaged region efficiently.

5. Experimental Results

The proposed algorithm implemented using a MATLAB program and tested using several images and video frames. To test the implemented algorithm, we utilized a Personnel Computer (PC) with Microsoft Windows 7 OS and 2.8GHZ Core Duo 2 CPU and 6MB cache.

TABLE I shows the results of text localization algorithm. In this table, the results of the proposed algorithm have been shown for different text languages including English, Arabic and Korean. In this table, the results of text localization algorithm based on Discrete

Cosine Transform (DCT) [29] are also shown for comparison. In this table, *NI, DB, FP, TP, FN, DR, FAR* and *FRR* stand for the Number of Images, Detected Blocks, False Positive, True Positive, False Negative, Detection Rate, False Acceptance Rate and False Rejection Rate respectively. *DR*, FAR and *FRR* are defined as follows:

$$DR = TP / (TP+FN) \quad (13)$$

$$FAR=FP/(TP+FN) \quad (14)$$

$$FRR=FN/(TP+FN) \quad (15)$$

As TABLE I shows the proposed algorithm is more efficient in detecting text blocks. Fig. 7 shows the results of the text extraction algorithm. The figure shows some candidate text blocks and the extracted characters. The results of Sauvola method [10] and Otsu method [30] are also shown for comparison. As the figure shows the proposed algorithm outperform the Sauvola and Otsu methods.

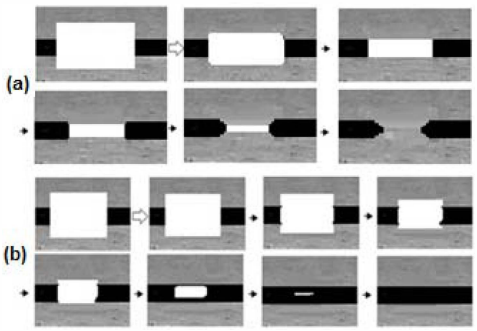

Fig. 6: The results of inpainting algorithm using texture matching (a) standard texture matching, (b) the proposed algorithm.

TABLE I: Results of Text Localization Algorithm

method	Language	NI	DB	FP	TP	FN	FAR	FRR	DR
Proposed method	English	40	55	5	50	2	0.096	0.038	96.1%
	Korea	50	91	9	82	4	0.104	0.046	86.3%
	Arabic	50	111	12	99	8	0.112	0.074	83.1%
DCT based method [29]	English	40	74	30	44	7	0.588	0.137	54.3%
	Korea	50	291	167	124	51	0.954	0.291	51.2%
	Arabic	50	195	70	125	16	0.496	0.113	59.2%

Fig. 8 shows the results of the proposed inpainting algorithm on several frames of a video file. As it is shown in the figure, the reconstructed area is not recognizable visually. To measure the distortion generated by the proposed inpainting algorithm, we added some texts blocks to about 100 video frames. Then after applying the proposed text localization, extraction and inpainting algorithm, the difference between the reconstructed and original image is used to measure the distortion of the inpainting algorithm.

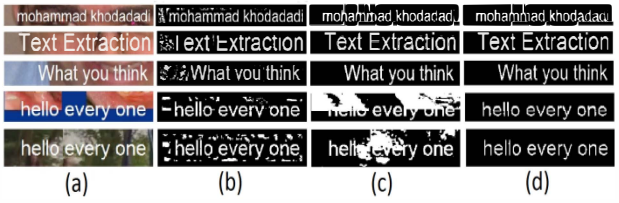

Fig. 7: The results of text extraction algorithm, (a) candidate text block, (b) extracted characters using Sauvola method [7,8], (c) extracted characters using Otsu method [30], (d) extracted texts using the proposed method.

Fig. 8: The results of text localization and inpainting algorithm (a) original images, (b) the extracted texts, (c) the reconstructed images after applying the proposed text extraction and inpainting algorithm.

We also compared the proposed inpainting algorithm with the Total Variational (TV) algorithm [12, 15] with 1000 iterations. Our method performs reconstruction in only one iteration.

To measure the efficiency of the inpainting algorithm, we utilize Peak Signal-to-Noise Ratio (PSNR), Mean Squared Error (MSE), Min Error (MinError) and Median Error (MedError) criterions as follows:

$$MSE=\frac{1}{3m\times n}\sqrt{\sum_{c=R,G,B}\sum_{j=0}^{n-1}\sum_{i=0}^{m-1}(I^c_{org}(i,j)-I^c_{reast}(i,j))^2} \quad (16)$$

$$MinError=\min_{c,i,j}\left(\left|I^c_{org}(i,j)-I^c_{reast}(i,j)\right|\right) \quad (17)$$

$$MedError=median_{c,i,j}\left(\left|I^c_{org}(i,j)-I^c_{reast}(i,j)\right|\right) \quad (18)$$

$$PSNR=20\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \quad (19)$$

Here m and n are image width and height respectively, I^c_{org} and I^c_{reast} are original and reconstructed images for color channel c respectively and MAX_I is the maximum possible pixel value of the image.

TABLE II shows the results of inpainting algorithms using the proposed and *TV* algorithm. The table shows that the distortion of the proposed algorithm is less than the *TV* algorithm.

TABLE II: Results of Inpainting Algorithm for the Proposed and TV Algorithm

Method	Error Parameter			
TV method [12,15]	MSE	MinError	MedError	PSNR
	0.015	0.000006	0.02	18.15
Our method	0.0082	0	0.02	20.87

5. Conclusions

In this paper a new method for text localization, extraction and inpainting was proposed. The text localization algorithm was based on image gradient, which proper text verification algorithms were used to localize the text blocks efficiently. Then we estimated color for text and background areas and used color segmentation algorithm to extract text characters precisely. The proposed inpainting algorithm is also a fast algorithm which needs only one iteration. The proposed algorithms were tested with different images and video frames and compared with other algorithms. The experimental results showed the efficiency of the proposed algorithms in text localization, extraction and inpainting stages.

References

[1] W. Fan, J. Sun, Y. Katsuyama, Y. Hotta and S. Naoi, "Text Detection in Images Based on Grayscale Decomposition and Stroke Extraction," *in Proc.* 2009 *IEEE Pattern Recognition Conf.,* pp. 1 - 4.

[2] D. Q. Zhang and F. H. Chang, "Learning to detect scene text using a higher-order MRF with belief propagation", in *Proc. 2004 of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops,* pp.101-107.

[3] J. Zhang, and R. Kasturi , " Text Detection Using Edge Gradient and Graph Spectrum ," *in Proc. 2010 International Conference on Pattern Recognition* , pp. 3979-3982.

[4] J. Gllavata, E. Qeli, and B. Freisleben, " Detecting Text in Videos Using Fuzzy Clustering Ensembles," *in Proc. 2006 8th IEEE International Symposium on Multimedia* , pp. 283-290.

[5] X. Zhao, K. H. Lin, Yun. Fu, Y. Hu, Y. Liu, and T. S. Huang, " Text From Corners: A Novel Approach to Detect Text and Caption in Videos," *IEEE Trans. Image Processing, vol.* 20, no. 3, March 2011.

[6] W. Kim and C. Kim,"A New Approach for Overlay Text Detection and Extraction From Complex Video Scene", *IEEE Trans. Image Processing, vol.* 18, no. 2, Feb. 2009.

[7] P. Shivakumara, T. Q. Phan, and C. L. Tan, "A Laplacian Approach to Multi-Oriented Text Detection in Video", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 33, no. 2, Feb. 2011.

[8] W. Niblack, *An introduction to digital image processing,* Prentice Hall, 1986, pp. 115–116.

[9] C. Wolf, J. Jolion, and F. Chassaing. "Text localization, enhancement and binarization in multimedia documents," *in Proc. 2002 International Conference on Pattern Recognition(ICPR)*, pp. 1037–1040.

[10] J. Sauvola and M. Pietikainen. "Adaptive document image binarization," *Pattern Recognition*, vol. 33, pp. 225–236, 2000.

[11] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballesterm, "Image Inpainting,",*in Proc. 2000 SIGGRAPH Conf.,* pp. pages 417-424.

[12] T. F. Chan and J. Shen, "Mathematical models of local non-texture inpaintings", *Society for Industrial and Applied Mathematics, vol.* 62, no. 3, pp. 1019–1043, 2002.

[13] T. F. Chan and J. Shen, "Non-texture inpainting by curvature-driven diffusions (CDD)". *Journal. Visual Comm. Image Rep*, vol. 12, no. 4, pp. 436–449, 2001.

[14] M.Bertalmio, " Strong-Continuation, Contrast-Invariant Inpainting With a Third-Order Optimal PDE", *IEEE Trans. Image Processing,* vol. 15, no. 7, July 2006.

[15] J. M. Fadili and G. Peyre, "Total Variation Projection With First Order Schemes," *IEEE Trans. Image Processing,* vol. 20, no. 3, March 2011.

[16] H. Guo, et al., "A structure-synthesis image inpainting algorithm based on morphological erosion operation" , *in Proc. 2008* International Congress on *Congress on Image and Signal Processing,* pp. 530 – 535.

[17] P. Elango and K. Murugesan, "Digital Image Inpainting Using Cellular Neural Network," *Int. Journal. Open Problems Compt. Math.*, vol. 2, 2009.

[18] J. Park, D. Park, R. J. Marks, and M. A. El-Sharkawi, "Recovery of Image Blocks Using the Method of Alternating Projections," *IEEE Trans. Image Processing,* vol. 14, no. 4, April 2005.

[19] D. Liu, X. Sun, F. Wu, S. Li, and Y. Q. Zhang, "Image Compression With Edge-Based Inpainting", *IEEE Trans. Circuits and Systems for Video Technology,* vol. 17, no. 10, Oct. 2007.

[20] Z. Xu and J. Sun, "Image Inpainting by Patch Propagation Using Patch Sparsity," *IEEE Trans. Image Processing, vol.* 19, pp. 1153-1165, 2010.

[21] Y. Wu, M. Wang and X. Liu "A Fast Inpainting Algorithm Using Half-Point Gradient," *in Proc.* 2009 *Second International Workshop on Computer Science and Engineering*, pp. 594-598.

[22] L. Demanet, et al., "Image inpainting by correspondence maps: a deterministic approach," *Applied and Computational Mathematics*, vol. 1100, pp. 217-5, 2003,.

[23] E. Pnevmatikakis and P. Maragos, "An inpainting system for automatic image structure-texture restoration with text removal," *in Proc. 2008 15th IEEE International Conference on Image Processing (ICIP2008)* , pp. 2616-2619.

[24] A.Criminisi, P.PErez, and K.Toyama, "Region Filling and Object Removal by Exemplar-Based Image Inpainting", *IEEE Trans. Image Processing,* vol. 13, no. 9, Sept. 2004.

[25] M.Bertalmio, L.Vese, G.Sapiro, "Simultaneous Structure and Texture Image Inpainting", *IEEE Trans. Image Processing,* vol. 12, no. 8, Aug. 2003.

[26] X. Li, W. Wang, Q. Huang, W. Gao , L. Qing, " A HYBRID TEXT SEGMENTATION APPROACH," *in Proc. 2009 IEEE International Conference on Multimedia and Expo,* pp. 510-513.

[27] M. N. Favorskaya, A. G. Zotin, and M. V. Damov, " Intelligent Inpainting System for Texture Reconstruction in Videos with Text Removal," *in Proc. 2010 International Congress on Ultra Modern Telecommunications and Control Systems and Workshops,* pp. 867-874.

[28] G. Miao, Q. Huang, S. Jiang, and W. Gao, "Coarse-to-Fine Vide Text Detection ," *in Proc.* 2008 *IEEE International Conference on Multimedia and Exp,* pp. 569-572.

[29] Z. Xu , H. Linging , F. Zou , Z. Lu , P. Li and T. Wang , "Fast and robust video copy detection scheme using full DCT coefficients," *in Proc. 2009 IEEE International Conference onMultimedia and Expo*, pp. 434 – 437.

[30] Y. HAO and F. ZHU, "Fast Algorithm for Two-dimensional Otsu Adaptive Threshold Algorithm ," Journal of Image and Graphics, vol. 4, 2005.