CIIT 2008 SCANNED IMAGES ENHANCEMENT

Tihomir Lazarov
BAS
Sofia, Bulgaria

Georgi Gluhchev
BAS
Sofia, Bulgaria

Abstract

When examining text materials often times documents are old, authentic, of rare kind or of bad quality. The access to some documents is many times limited because of the state and kind of the materials. The researchers who examine texts need the sources in a handy format, which can improve the process of exploring the data. With the development of technologies this is possible if the materials are digitized – scanned, shot with a camera, etc. Although images are more comfortable to work with they take a significant amount of disk space. If the text from the source material is put in a plain text file it would take tenths of times less space. Creating the text files manually from the source materials is a time consuming and hard work. Automating that process is a challenging work for many scientists. Today we have lots of OCR (Optical Character Recognition) software which can recognize characters from digitized materials to some extent. However there are plenty of cases when the image is of too bad quality which makes it almost impossible for the tool to recognize the characters. Often the text is hardly read, there are shadows from the scanning process, there are places where the page has been burnt, the paper is somehow transparent and the back side content is slightly visible, and so on. Image quality enhancement is needed to improve the readability of the digitized source data and to ease the process of OCR. This work describes some methods and techniques for enhancement of images containing text so that tools for OCR can have a clean source to work on. There are two basic phases which take place in the process: contrast enhancement and noise filtering; separating the objects from the background for easier distinguishing between objects.

Introduction

Often experiments to extract the textual data from a raster image lead to failure, because of the quality of the source image. The more the physical source is of bad quality (smears, paper opacity, shadows, etc.), the more the raster bitmap is of worse quality. The quality of the bitmap depends on the technical tools which are used during the digitalization of the source. Overcoming the quality barrier eases the quality of extraction of the textual information and increases the possibility to make the extraction automated.

The preliminary processing of an image is needed in the cases which one may gain more information about the objects on it or to emphasize on details of interest on the image. The phases the processing goes are:

- improving the contrast of the image (if needed);
- noise filtering;
- separating the objects from the background – before we proceed with that step the image contains clearly distinctive objects. The background is usually what has to be eliminated as the objects are of more interest;

General

I. Image Preprocessing

A. Contrast Enhancement

1) Histogram Normalization

When the image is too bright or too dark preprocessing is needed to darken or respectively lighten it. This way objects become more notable.

An image which is too bright or too dark has a histogram which columns are either clustered on the one side or on the other. The columns need to be redistributed between the minimal and the maximal values of the histogram bars. The difference between these values is called an amplitude. The value of each column is recalculated as , where X is the value of the column; A is the amplitude; V is the number of gray scale nuances (in this work it is 256).

This way the columns of the histogram become more equally varying between 0 and 255 than before where there were too low or too high columns.

B. Noise Filtering

Usually in a raster image there is a significant amount of pixels which are not part of the original background, but present there because of the imperfections in the technologies used for digitalization of objects from the real world. These pixels differ dramatically from those in their vicinity. They are called noise. This property of the noise pixels is used in the following methods engaged in their removal.

1) Mean Filter

This algorithm is relatively simple. It is based on finding the mean value of the intensity of a vicinity with odd number of pixels. This value is set as the intensity of the pixel in the center of the region. This way the sharp differences between pixels from an ideal noiseless image and the noise pixels are eliminated. After that filtering the image becomes a little blurry, but the noise is relatively well removed.

The advantage of this method is that it's fast and easy for implementation. The disadvantage is the very blurring of the bitmap and the extra pixel intensity values which are added.

The image matrix is traversed consequently by columns for every row with a mask matrix of odd number of cells. The number of cells in the mask matrix is less than the number of cells in the image. Usually its size is of 3x3 or 5x5. On each step the average value of the intensities under the masking matrix is calculated. It is set as an intensity value of the pixel which is beneath the center pixel from the masking matrix. To avoid overlapping of the old and new values the changes are applied over a copy of the original image. Another common problem is the calculation of the new values of the pixels from the border of the image. To deal with that problem the image is added an extra frame with width equals to the half of the mask matrix width. Usually the color of the border is black or white, but it's best to duplicate the colors of the original image border pixels.

254	254	254
251	~~160~~ 215	254
250	254	254

Figure 1: Mean filter

2) Median Filter

This algorithm again uses a region of odd number of pixels. Now the values of the intensities of pixels in the region are sorted and the middle element is taken. Its value is set as the value of the pixel in the center of the region. This way there are no artificially added values as in the mean filter, but they are chosen out of existing values. In this case there's no so much image blurring, but we have a certain darkening (in case the background is light and the noise is dark points) or lightening of the image (in case the background is dark and the noise is lighter points).

The matrix of the original image is traversed with a filtering matrix of an odd number of cells. On every step the intensity values below the filtering matrix are sorted. The middle element's value from the sorted list is set as an intensity value of the pixel under the middle pixel from the filtering matrix. Again the changes are made over a replica of the original image which has an artificially extended border as described in the previous algorithm.

254	254	254
251	~~160~~ 254	254
250	254	254

Figure 2: Median filter matrix sorted elements -

160, 250, 251, 254, **254**, 254, 254, 254, 254

3) Filtering With A Matrix Mask

This algorithm is universal, because it uses one and the same operation over the image which works with different kinds of operands. The operands in that case are matrix masks or matrices. The different values in the matrix cells make the operations to have different effect over the image. It's easy to see that the mean filter is a private case of filtering with a matrix mask with elements 1/N, where N is the matrix dimension. When doing a noise removal usually 3x3 matrices of the following kind are used:

1/10	1/10	1/10
1/10	2/10	1/10
1/10	1/10	1/10

Figure 3a)

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16

Figure 3b)

1/8	1/8	1/8
1/8	0	1/8
1/8	1/8	1/8

Figure 3c)

The matrix goes over the image. On every step a value is calculated as a sum of the products of the values of the matrix cells and their respective cells on the image. The resulted value is set as the value of the pixel below the middle cell of the matrix mask.

$$X_{i,j} = \sum_{m=i-H/2}^{i+H/2} \sum_{n=j-W/2}^{j+W/2} F_{m-i+H/2,n-j+W/2} A_{m,n} \quad (1),$$

where

- $X_{i,j}$ is the intensity value of the pixel below the matrix center pixel; i and j are indexes of pixels from the original image;
- W, H – respectively the width and height of the filtering matrix;
- $F_{p,q}$ – the value of the cell (p, q) from the filtering matrix;
- $A_{m,n}$ – the intensity of the pixel on position (m, n) from the original image;

II. Separating Objects From Background

1) Otsu Algorithm For Histogram Thresholding

The Otsu algorithm finds threshold in an image histogram which is used to separate objects from the background. All pixels which represent the part of the histogram from one side of the thresholding value are assumed as points from the foreground, the others – from the background. The algorithm goes through two phases – threshold estimating; changing the histogram of the image so that points representing the background to be one color, while points from the objects to be set with another color. The result is a monochrome bitmap.

To find the threshold image histogram is generated. It is an array of elements (also called *columns* or *bars)*. The number of elements represents the number of different intensity values the image may have. In this case they are from 0 to 255. Each value represents the number of pixels from the image which have the same intensity as the value of the index of the *bar*. For every nuance of the gray color is set a classic probability p_i which is F[i] / N, where F[i] is the number of pixels in the *i*-th nuance; *N* is the number of all pixels in the image. The histogram (the array of values) is divided into two groups – 0..*t*; *t* + 1..*L*-1, where *L* is the number of elements in the histogram (L = 256 in our case).

The *t* parameter goes from 0 to 255. The *t* between the two groups is calculated the following way:

$$t = \omega_1(m_1 - M)^2 + \omega_2(m_2 - M)^2, \quad (2)$$

where

- $\omega_{1, 2}$ – the sum of the probabilities respectively from the left and right groups;
- $m_{1, 2}$ – the mean value of the grey nuance from the left and right groups respectively; it is calculated by the formula:

$$m_k = \sum i \frac{p_i}{\omega_i} \quad (3)$$

- M – the mean value of the whole histogram calculated by:

$$M = \omega_1 m_1 + \omega_2 m_2 \quad (4)$$

The maximal value *t* is the desired threshold. All values from the histogram which are less than *t* are set to 0 and all that are greater – to 255. Then the image has pixels of two colors – black and white which make it easier for objects to separate from the background.

2) Algorithm For Separating Objects From The Background Using Image Scaling

This algorithm works well in cases when objects are of size similar to text on a page while the background pattern is either flat or constant or has significant regions with the same pattern.

The very name of the algorithm gives a hint that scaling will be the essential phase of the operation. Scaling of an image is defined as proportional shrinking or stretching of the bitmap. The coefficient for shrinking and stretching is chosen empirically. When shrinking an image there is a certain loss of pixel information which is used as an advantage.

The shrinking makes the objects to loose their clarity to some extent, even some of them may disappear. When stretching back to the original dimensions the objects are almost merged with the background. This way the product is an image which is quite close to an image representing the background of the original bitmap without the objects.

The scaling coefficient is chosen as 20% of the width and respectively height of the image. When the coefficient is greater the objects and the background degrade so much that when scaling back the background doesn't have much in common with the original image background. If the coefficient is very small the objects are not well removed from the background.

After shrinking of the image noise filtering is applied using any of the masking matrices mentioned before. This way blurring effect is enforced. The image is then scaled back to the original dimensions. Mean filter is applied. At that point the objects are quite blurred so they can be assumed as part of the background. This method is often referred to as *background approximation*. The approximated background is subtracted from the original image and the result contains objects which are more distinctive than the background which is almost removed. The object edges may be slightly thicker, because blurring effect was applied before. The Otsu algorithm is applied so that the result may be a monochrome bitmap. Morphological erosion helps thinning the object edges. The original image is divided into non-overlapping rectangles and on each of them Otsu is applied. "Conjunction" is applied between the eroded image and the Otsu-ed original image.

Here are the algorithm steps in more technical details:

1. The original image (*image_original*) is scaled down with 20% by each of its dimensions. During the shrinking each 5th pixel from the row or the column is taken only. This results in pixel loss, but this is the desired effect as background approximation is the goal;
2. Noise filtering is applied with a 3x3 matrix which has all its elements set to 1/8 but the center one which is 0;
3. The shrunk image is expanded back 5 times. The result has the dimensions of the original image. During the stretching there are regions with "holes" which are filled with the average value of a rectangular vicinity centered in the "hole" pixel which has at least one non-hole pixel;
4. Mean filter is applied using a 21x21 matrix. The result image is *image_modified*.
5. The original image (*image_original*) is subtracted from the result (*image_modified*). The operation *subtraction* is defined as subtracting the respective components of each respective pixel. After the operation some component values may be negative. They are replaced with 255 + the_negative_value. The result is *image_subtracted*.
6. Otsu is applied over *image_subtracted*. The result is *image_otsu_subtracted*.
7. As with scaling and mean filtering there is some contour blurring so the subtraction introduces border thickening. Morphological erosion is applied to make it thinner. Erosion is applied using structural element with 3x3 dimensions which has zeros in the corners and ones in the other positions. The result is *image_otsu_subtracted_erosion*.
8. The original image is divided into non-overlapping rectangles with size of 100x100. Otsu algorithm is applied over each one of them. The result is *image_rect_otsu*.
9. "Conjunction" operation is applied between *image_otsu_subtracted_erosion* and *image_rect_otsu*. This operation is applied between the respective components of the respective pixels from the bitmaps

(as they have the same dimensions). As both images have passed the Otsu algorithm they have pixels with 0 and 255 values. The "conjunctions" is defined as follows: if the values of the respective components are 0, the result is 0. Otherwise it's 255.

III. Experimental Results

1) Used Technologies And Methods

The application that is used for image processing is written in "C" language. It uses the ImageMagick library only for loading an input image and to store images as an enhanced output. A set of libraries is developed:

- Error handling;
- Loading, reading, writing image data – ImageMagick library is used for reading and writing images. The rest of the image operations are manually implemented – scaling, pixel manipulation;
- Morphological operations over images – functions for dilation, erosion, opening, closing;
- Working with histograms:
  - Otsu algorithm for finding optimal thresholds for component segmentation;
  - Histogram normalization;
  - Nonadaptive method for image contrast enhancement;
- Image filtering – implemented functions for
  - converting a color bitmap into a grayscale one;
  - algorithms for noise filtering with matrix masks (median filter, mean filter);
  - algorithms for histogram altering;
  - algorithm for separation of objects from background using scaling;

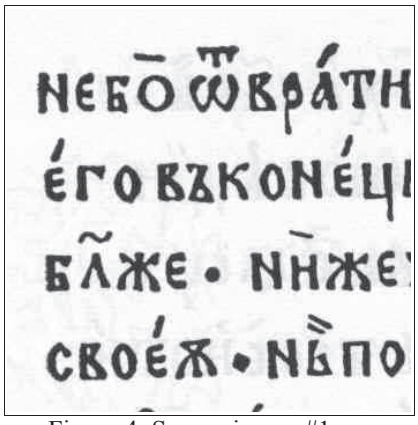

Figure 4: Source image #1

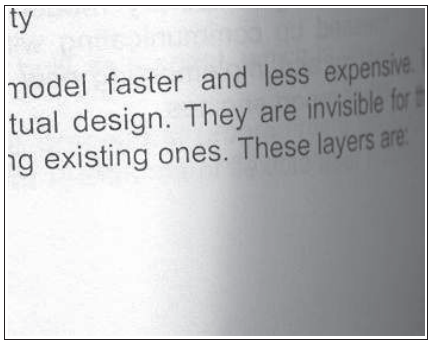

Figure 5: Source Image #2

2) Description Of Experiments

A combination of algorithms is used to achieve the desired result – an image with cleaned noise and clearly distinctive objects from the background.

Each of the experiments is applied over two test images. Each of the combination of operations has several basic steps – noise removal; separation of objects from the background. Sometimes as an additional step is applied morphological operation to improve the result.

Test 1

This test represents the most natural approach to the problem – noise cleanup, applying the classical Otsu algorithm and morphological "opening". This experiment works well with images which has clearly distinctive constant background with almost no variations and well formed objects separated from the background. It doesn't work well with image which has shadows which have appeared during scanning. Here are the algorithm operations:

1. Noise removal using filtering mask
   1 1 1
   1 0 1
   1 1 1
2. Otsu algorithm over a histogram of the full image;
3. Application of morphological "opening" with the structural element:
   0 1 0
   1 **1** 1
   0 1 0

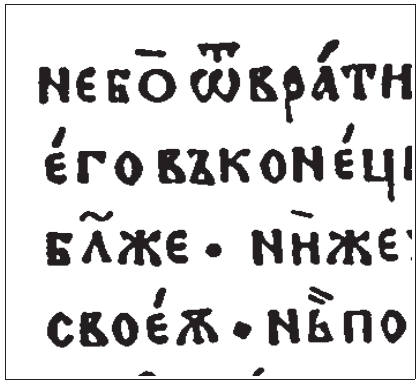

Figure 6: Sample Image #1 Result

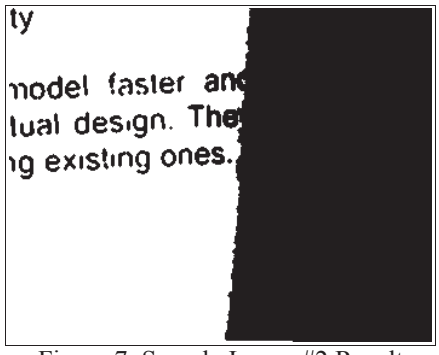

Figure 7: Sample Image #2 Result

Test 2

The experiment is similar to the previous one with the difference that Otsu algorithm is applied over non overlapping rectangles which the image is divided on. Again with images having a constant background the result is better. The test gives better results over images with differences in the background, but there are still artifacts at the places near by the borders of the rectangles (during the Otsu algorithm). The steps:

4. Mean algorithm filtering;
5. Otsu algorithm over each of the non-overlapping rectangles;
6. Application of morphological "closing"

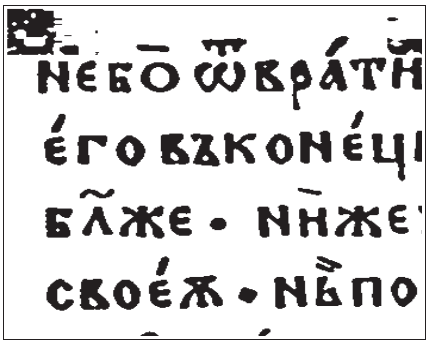

Figure 8: Source Image #1 Result

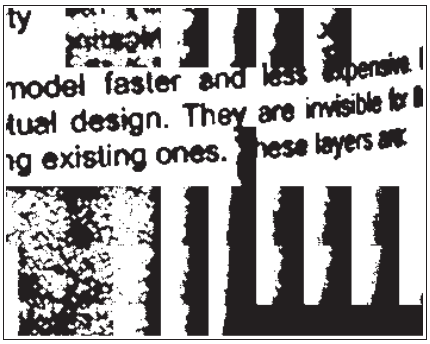

Figure 9: Source Image #2 Result

Test 3

This test gives the best results so far, because it well adapts to the background so that extracting the objects becomes an easier operation. It simply applies the algorithm for separation of objects and background using scaling (described above).

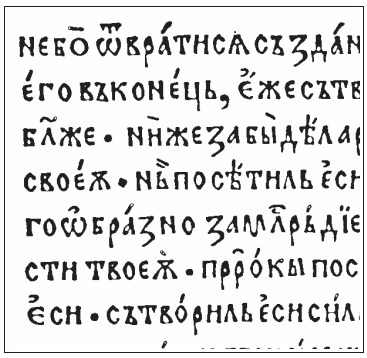

Figure 10: Source Image #1 Result

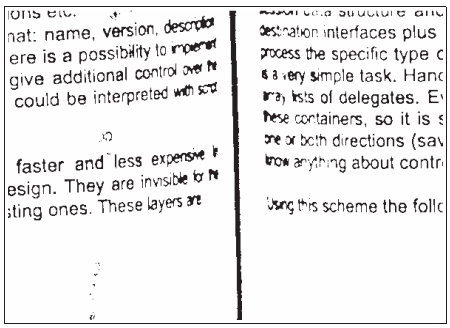

Figure 11: Source Image #2 Result

References

[1] Journal Of Information Science And Engineering 17, 713-727 (2001), A Fast Algorithm for Multilevel Thresholding

[2] Pratt, William K., Digital Image Processing, third edition, John Wiley & Sons, Inc., 2001, ISBN 0-471-37407-5

[3] Dougherty, Edward R., An Introduction to Morphological Image Processing, The Society of Photo-Optical Instrumentation Engineers, 1992, ISBN 0-8194-0845-x

[4] Oliver A Nina, William A Barrett, Thresholding of Text Documents - http://www.fht.byu.edu/prev_workshops/workshop07/slides/2/Thresholding-Text-Documents.pdf