Expert Systems with Applications 39 (2012) 4032–4043

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Display text segmentation after learning best-fitted OCR binarization parameters

Antonio Fernández-Caballero [a,b,*], María T. López [a,c], José Carlos Castillo [a]

[a] Universidad de Castilla-La Mancha, Instituto de Investigación en Informática de Albacete (I3A), 02071 Albacete, Spain
[b] Universidad de Castilla-La Mancha, Escuela de Ingenieros Industriales de Albacete, Departamento de Sistemas Informáticos, 02071 Albacete, Spain
[c] Universidad de Castilla-La Mancha, Escuela Superior de Ingeniería Informática, Departamento de Sistemas Informáticos, 02071 Albacete, Spain

ARTICLE INFO

Keywords:
Optical character recognition
Text segmentation
Binarization
Learning

ABSTRACT

In this paper text segmentation in generic displays is proposed through learning the best binarization values for a commercial optical character recognition (OCR) system. The commercial OCR is briefly introduced as well as the parameters that affect the binarization for improving the classification scores. The purpose of this work is to provide the capability to automatically evaluate standard textual display information, so that tasks that involve visual user verification can be performed without human intervention. The problem to be solved is to recognize text characters that appear on the display, as well as the color of the characters' foreground and background. The paper introduces how the thresholds are learnt through: (a) selecting lightness or hue component of a color input cell, (b) enhancing the bitmaps' quality, and (c) calculating the segmentation threshold range for this cell. Then, starting from the threshold ranges learnt at each display cell, the best threshold for each cell is gotten. The input and output data sets for testing the algorithms proposed are described, as well as the analysis of the results obtained.

1. Introduction

With the rapid progress of digital imaging acquisition techniques, text recognition becomes increasingly important. The problems in text recognition seem mainly to be due to segmentation (Aas & Eikvil, 1996; Jain & Bhattacharjee, 1992; Montazer, Saremi, & Khatibi, 2010). The problem of text segmentation lies in effective separation into character foreground and background (e.g. Shapiro, Gluhchev, & Dimov, 2006). In this sense, most optical character recognition (OCR) methods assume that individual characters can be isolated.

For digital character images, traditional OCR methods do not provide satisfactory recognition performance because it is very difficult to get clear binary character images (Liu, Wang, & Dai, 2006). For example, the performance of OCR can be degraded if the characters in binarized images are broken or blurred. There are many factors that affect the performance of binarization algorithm, such as complex signal-dependent noise (Zheng, Li, & Doermann, 2004) and variable background intensity, which are caused by non-uniform illumination, shadow, smear, smudge or low contrast (Huang, Ahmadi, & Sid-Ahmed, 2008). In recent decades, some research has been concentrated on character recognition, which can be divided into two categories: extraction of the character features from the binary character image by advanced binarization (Kanungo, Haralick, Baird, Stuezle, & Madigan, 2000; Taylor & Dance, 1998), and direct extraction of the features from the gray scale image (Lee & Kim, 1995; Wang & Pavlidis, 1993). Bi-thresholding or binarization is probably the most simple but effective technique for text segmentation. Therefore it has been intensively used for more the last three decades. Indeed, the problem of text binarization may be considered as a pre-processing step for OCR (Gupta, Jacobson, & Garcia, 2007). The binarization methods reported in the literature can be categorized into (1) histogram-based methods (Prewitt & Mendelsohn, 1965), (2) clustering-based methods (Otsu, 1979), (3) object attribute-based methods (Pikaz & Averbuch, 1996), and (4) discrimination based on local pixel's characteristics (Yan & Wu, 1994).

In the clustering-based methods, the gray-level samples are usually clustered in two parts as background and foreground. The Otsu method (Otsu, 1979) is the most referenced thresholding method. In this method the weighted sum of within-class variances of the foreground and background pixels are minimized to establish an optimum threshold. This method gives satisfactory results when the numbers of pixels in each class are similar. Additional methods include iterative thresholding (Yanni & Horne, 1994), minimum error thresholding (Cho, Haralick, & Yi, 1989; Kittler & Illingworth, 1986), as well as fuzzy clustering thresholding (Jawahar, Biswas, & Ray, 1997). For local pixels adaptive algorithms, a threshold is calculated at each pixel, which depends on some local statistics like range, variance, or surface-fitting parameters of the neighboring pixels. A surface fitted to the gray-level

* Corresponding author at: Universidad de Castilla-La Mancha, Instituto de Investigación en Informática de Albacete (I3A), 02071 Albacete, Spain. Tel.: +34 967 599200; fax: +34 967 599224.
E-mail address: caballer@dsi.uclm.es (A. Fernández-Caballero).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.09.162

landscape is used as a local threshold in Yanowitz and Bruckstein (1989), Shen and Ip (1997). Also traditional neural networks (Yan & Wu, 1994) are introduced to discriminate the pixels into background and foreground, according to the characteristics around every pixel. In another paper (Yan, Zhang, & Kube, 2005), a general locally adaptive thresholding method using neighborhood processing is presented. The method makes use of local image statistics of mean and variance within a variable neighborhood and two thresholds obtained from the global intensity distribution.

Recently the application of dynamic Bayesian networks (DBN) to the recognition of noisy characters has been investigated (Likforman-Sulem & Sigelle, 2008). Similar stochastic approaches such as hidden Markov models (HMM) have been widely applied to text recognition (Schenkel & Jabri, 1998). This is largely due to their ability to cope with incomplete information and non-linear distortions. HMM may also be used as classifiers for single characters (Park & Lee, 1998). A recent paper (Liu et al., 2006) proposes an approach for the low resolution character recognition, which fits the input character for the appropriate character database according to the input image quality. It is composed of character image quality evaluation and character recognition.

It should be mentioned that most of the algorithms are based on the hypothesis that the foreground is much darker or clearer than the background, which is true in most of the cases (Antani, Crandall, & Kasturi, 2000). Though much effort has been devoted to the binarization, extraction of characters from noisy images is still a big challenge (Huang et al., 2008). Text segmentation methods assume that the gray-scale distribution is bimodal and that characters a priori correspond to either the white part or the black part, but without providing a way of choosing which of the two possibilities applies. Great efforts are thus devoted to performing better binarization, combining global and local thresholding (Kamada & Fujimoto, 1999), M-estimation (Hori, 1999), or simple smoothing (Wu, Manmatha, & Riseman, 1997). However, these methods are unable to filter out background regions with similar gray-scale values to the characters. If the character gray-scale value is known, text enhancement methods can help the binarization process (Sato, Kanade, Hughes, & Smith, 1998). However, without proper estimation of the character scale, the designed filters cannot enhance character strokes with different thickness (Chen, Shearer, & Bourlard, 2001). So, in Sato et al. (1998) a linear interpolation technique to magnify small text at a higher resolution is used. Text regions are detected and localized using Smith and Kanade's method (Smith & Kanade, 1995), and sub-pixel linear interpolation is applied to obtain higher resolution images. And, in Li, Doermann, and Kia (2000), and Li and Doermann (2000) several approaches for text enhancement have been proposed. For example, they use the Shannon interpolation technique to enhance the image resolution of video images. The image resolution is increased using an extension of the Nyquist sampling theorem and it is determined whether the text is normal or inverse by comparing it with a global threshold and background color.

When the captured images include colors, the text segmentation problem increases significantly, leading sometimes to a color reduction (Makridis, Nikolaou, & Papamarkos, 2010). In Wang and Kangas (2003), color clustering is used to separate the color image into homogeneous color layers. Next, for each color layer, every connected component in color layers is analyzed using black adjacency graph (BAG), and the component-bounding box is computed. Then, for coarse detection of characters, an aligning-and-merging-analysis (AMA) scheme is proposed to locate all the potential characters using the information about the bounding boxes of connected components in all color layers. Finally, to eliminate false characters, a four-step identification of characters is used. Also, a binarization technique of characters in color using genetic algorithms (GA) to search for an optimal sequence of filters through a filter bank has been proposed (Kohmura & Wakahara, 2006). The filter bank contains simple image processing filters as applied to one of the RGB color planes and logical/arithmetic operations between two color planes. Different adaptive binarization methods can for instance be (Yanowitz & Bruckstein, 1989); and in Trier and Taxt (1995) several methods are compared with respect to the obtained recognition results. The intrinsic characteristics of the text by using the stroke filter has also been considered (Jung, Liu, & Kim, 2008). First, the stroke filter briefly based on local region analysis is described. Second, the determination of text color polarity and local region growing procedures are performed successively based on the response of the stroke filter. Finally, the feedback procedure by the recognition score from an OCR module is used to improve the performance of text segmentation. In Tsai and Lee (2002) the binarization of color images by using the lightness and saturation features, components of the HLS (hue–lightness–saturation) color space is proposed. The binarization results are shown using only lightness, only saturation, or a combination of both lightness and saturation.

In this paper we propose to learn the best-fitted OCR binarization parameters for a commercial OCR library. This will make the segmentation have the highest probability of being the most correct. In Section 2, a general description of the approach is provided. Sections 3 and 4 introduce some pre-processing steps needed before any recognition algorithm. Then, Sections 5 and 6 introduce the method for learning the colors present in the display and the threshold ranges per cell, which enable a good performance when recognizing the characters. Afterwards, the parameters learnt previously are used to determine the optimal recognition values in Section 7. The data used to validate the approach, as well as the results obtained, are discussed in Section 8. Lastly, in Section 9, we offer our conclusions.

2. General description of the approach

The purpose of this work is to provide the capability to automatically evaluate standard display information, so that tasks that involve visual user verification can be performed without human intervention. The problem to be solved is to recognize text characters that appear on the display, as well as, the color of the characters' foreground and background. Our approach, as in most other previous works, is based on an accurate binarization of the text characters. But, camera-captured images often exhibit non-uniform brightness because it is difficult to control the imaging environment, unlike in the case of the scanner. The histogram of such images is generally not bi-modal and a single threshold can never yield an accurate binary document image. As such, global binarization methods are not suitable for camera images. However, most commercial OCRs use binarized images or gray-scale images as input.

In applications where you also want to detect the color (character or background color), camera-captured colored images will obviously have to be converted into gray scale or directly binarized. In any case, the basic parameter chosen to separate the foreground from the background is precisely the optimal threshold value, θ.

Moreover, our solution establishes that the characters have to fall into a pre-defined set of cells (or bitmaps) of the display. The display is divided into a pre-defined number of rows and columns. Each cell or bitmap in the RGB (red–green–blue) color space, $B_{RGB}(i,j)$, contains a single character, $char(i,j)$, where (i,j) is the coordinate of the cell's row and column. Also, each cell will have a height, h, and a width, w in the coordinates space $[x,y]$, where $x \in [1..w]$ and $y \in [1..h]$. Thus, the objective is to accurately recognize the ASCII values of each character, $char(i,j)$, contained in its

corresponding bitmap, $B_{RGB}(i,j)$. To obtain the character code, the solution proposed entails performing two phases. Training is proposed for the first phase in order to obtain an optimal threshold, $\theta_{opt}(i,j)$, for every input image cell. This optimal threshold will provide the best accuracy rate in commercial OCR module recognition. The second phase, so-called detection phase, uses the results from the previous phase, training phase, to obtain the character value.

Finally, please notice that the commercial OCR recognition module used in this work is the OCR provided by the Matrox company through the Matrox Imaging Library (MIL). The MIL Optical Character Recognition module reads and verifies mechanically generated character strings in 8-bit gray scale images, and provides results such as quality scores and validity flags. Functions are available to create, save, restore, modify, and inquire about fonts. The module also allows you to calibrate fonts, define search constraints, and read/verify character strings in the target image. The font information can be saved in a MIL-compatible (.mfo) font file format that can be used with other Matrox Imaging software products. The functions from the MIL library used are *MocrReadString*, which reads an unknown string from the specified target image using the specified OCR font context, and function *MocrGetResult*, which gets the specified type of result(s) from an OCR result buffer.

```
void MocrReadString (
MIL_ID ImageBufId,
MIL_ID FontId,
MIL_ID OcrResultId
)
```

where *ImageBufId* specifies the identifier of the image which contains the string to be read, *FontId* specifies the identifier of the OCR font context to be used to read the string from the target image, and *OcrResultId* specifies the OCR result buffer in which to place results of the read operation. All existing font controls and constraints are taken into account, and results are stored in the specified result buffer.

```
void MocrGetResult (
MIL_ID OcrResultId,
double ResultType,
void *ResultPtr
)
```

where *OcrResultId* specifies the identifier of the OCR result buffer from which to retrieve results, *ResultType* specifies the type of result(s) to be retrieved, and *ResultPtr* specifies the address in which to write the results. Now, the following values are available to retrieve results from a read or a verify operation:

- *M_TEXT*: Retrieves the entire text. This includes all available strings, their separators, and the terminating character.
- *M_TEXT_LENGTH*: Retrieves the total number of characters in the entire text.
- *M_TEXT_SCORE*: Retrieves the match score for the entire text as determined during the read/verify operation.
- *M_THRESHOLD*: Retrieves the value used to binarize the target image.

We are specifically concerned with the read character and score for the output results, since in this proposal, we well read the display characters one by one.

3. Image calibration

As told before, one of the greatest difficulties for an optimal segmentation in fixed positions of a textual display is the calculation of the exact starting and ending positions of each bitmap in the coordinate system of the display. This it is an exciting challenge, as important screen deformations appear due to the camera lens used for the display acquisition process. These deformations consist of a *"ballooning"* of the image, trimmed in the point to which the camera focuses. For this reason, it is essential to initially perform a calibration of the image. Let us remind, once again, that the segmentation in this type of displays is essentially based in an efficient bitmaps localization. It is absolutely mandatory to scan any captured image with no swelling up, row by row, or column by column, to obtain the precise position of each bitmap ($B_{RGB}(i,j)$). On the contrary, pixels of a given row or column might belong to an adjacent bitmap.

In order to solve this problem, a "dots grid", G_{dots}, is used as a pattern (see Fig. 1a). Each grid dot corresponds to the central pixel of a bitmap (or cell) $B_{RGB}(i,j)$ of the display. Once the grid points have been captured by the camera, the image ballooning and each dot deviation with respect to the others may be studied (see Fig. 1b).

Thanks to this information, and by applying the piecewise linear interpolation calibration method (Faugeras, 1993; Tsai, 1987), any input image, I, is "*de-ballooned*". Thus, this swelling up is eliminated, providing a resulting new image I_P. The centers of the dots are used to perform the calculations necessary to regenerate the

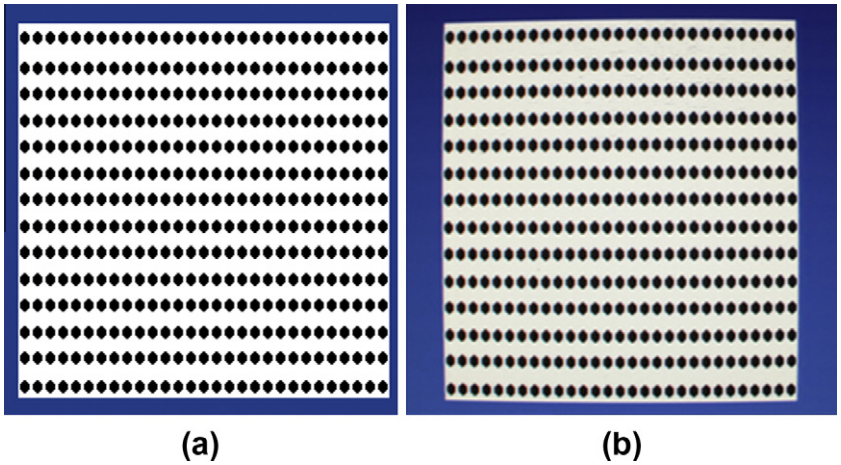

Fig. 1. (a) Optimal dots grid. (b) Captured dots grid.

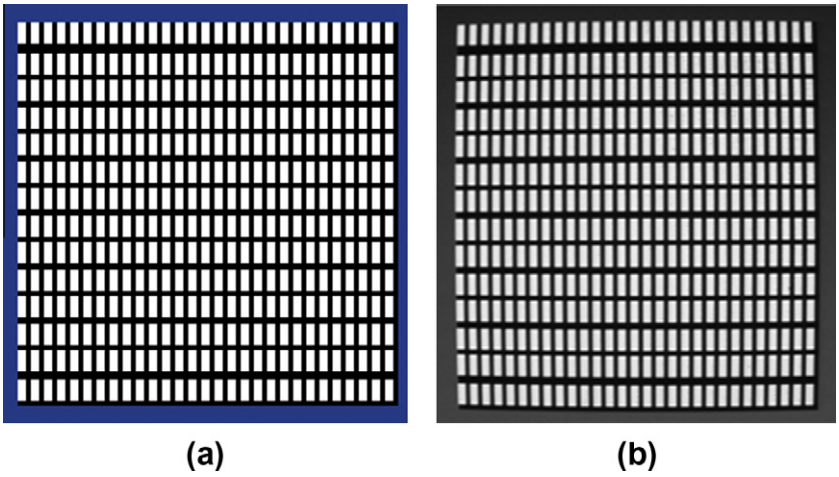

Fig. 2. (a) Optimal bitmaps grid. (b) Captured bitmaps grid.

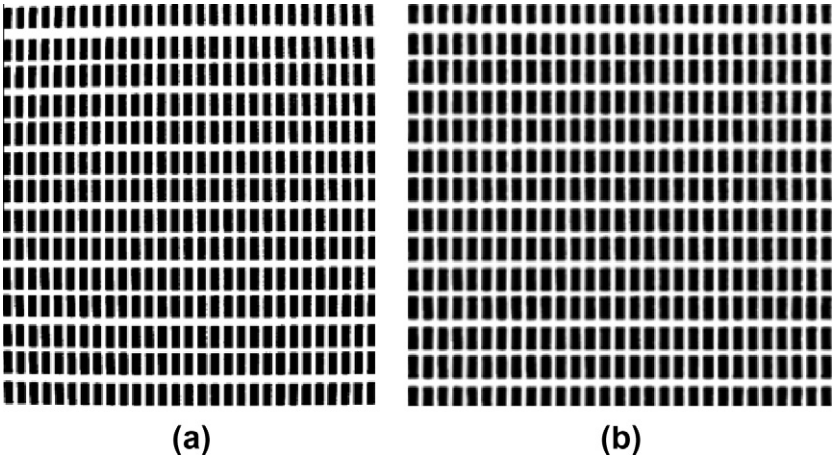

Fig. 3. (a) Binarized bitmaps grid. (b) Binarized and calibrated bitmaps grid.

original rectangular form of the input image. In addition, the average, $\overline{G_{dots}}$, of a certain number of captured dots grids is used as input to the calibration method to augment the precision of the process.

4. Bitmap localization

After calibration, the algorithms for *bitmap localization* are started. This phase is in charge of obtaining the most accurate localization of all bitmaps present in the calibrated image I_P. In other words, the algorithm obtains, for each bitmap $B_{RGB}(i,j)$ its initial and final pixels' exact positions. From the previous positions, also the bitmap's height, h, and width, w are calculated.

For performing the precise bitmap localization, another template (or pattern) is built up. This template consists of a "bitmaps grid" (see Fig. 2a), that is to say, a grid establishing the limits (borders) of each bitmap. The process consists in capturing this "bitmaps grid", G_{cells}, which, obviously, also appears convex after camera capture (see Fig. 2b). Again, a mean template image, $\overline{G_{cells}}$, is formed by merging a determined number of bitmaps grids captures. This process is driven to reduce noise that appears when using a single capture.

On the resulting average image, $\overline{G_{cells}}$, a series of image enhancement techniques are applied. In first place, a binarization takes place to clearly separate the background from the foreground (see Fig. 3a). The binarization is performed as shown in formula (1).

$$BG_{cells} = \begin{cases} 0, & \text{if } \overline{G_{cells}} \leqslant 135 \\ 255, & \text{otherwise} \end{cases} \quad (1)$$

Next, the calibration algorithm is applied to the bitmaps grid (see Fig. 3b), similarly to the calibration performed on the dots grid, in order to correct the distortion caused by the camera lens.

Once the template has been calibrated, it is now the time to perform little refinements on the bitmaps. For this purpose, an object search algorithm is used in the captured image. It is necessary to eliminate possible spots that do not represent bitmap zones. For this, a filter to eliminate too small or too big "objects" is applied. Then, the generated "objects" are analyzed. It is verified that the total number of "objects" corresponds with the total number of bitmaps in the display (that is to say, in the template). If this is the case, the resulting "objects" are sorted from left to right and from top to bottom.

While the position of the camera or the display type do not change during the segmentation process, the calibration and localization remain for all the searches in bitmaps. Nonetheless, some problems may arise during these phases. For instance, the camera may not be correctly adjusted. In this case, the processing of the cells fails irremediably. Some cells may appear united due to a

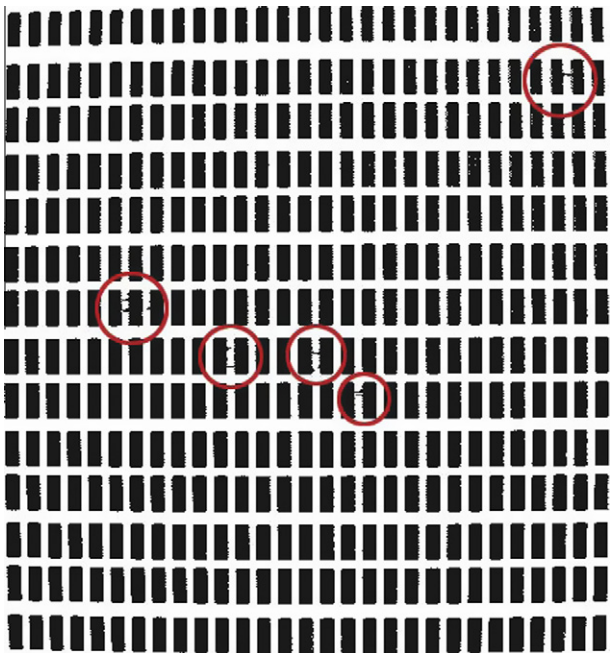

Fig. 4. Captured bitmaps grid after binarization in case of de-focusing.

sub-exposure (iris too much closed) or a de-focusing (see Fig. 4), or they disappear due to an over-exposure (iris too much open). Then, the localization function is unable to position the bitmaps appropriately, and, hence, to get their sizes. So, it is necessary to correctly adjust the camera lens and to repeat the complete process of calibrating the image and locating the bitmaps if any trouble occurs.

5. Learning the colors

An essential part of the processing of the characteristics of a bitmap is to obtain the background and character colors. Therefore, learning the colors consists of clearly determining the foreground and background colors of each bitmap. Please notice that in textual displays, it is not common to use a broad color spectrum but to work with a basic set of up to 16 colors.

To clearly discriminate background and character colors within a bitmap, the recommendations of the *World Wide Web Consortium* (*W3C*), were followed, using the *contrast ratio technique* (W3C, 2008). This technique is restrictive enough to provide the desired high contrast discrimination between pairs of colors (see Table 1). Contrast ratio, *cr*, is based on the calculation of the relative luminance *l* of the colors. Once we have obtained the luminance of two colors (l_1 and l_2) to be differentiated, the existing relation between them is calculated. If said relation exceeds a threshold of 5 set by the *W3C*, the background/character combination can be considered easily legible.

5.1. Assigning predefined basic colors to bitmap pixels

Firstly, each bitmap pixel is assigned the value of one of a set of predefined basic colors, as provided by the *W3C* consortium. Given the (*R*, *G*, *B*) components of a certain color, its relative luminance, *l*, is calculated from the following algorithm:

$$l = 0.2126 \times r + 0.7152 \times g + 0.0722 \times b \tag{2}$$

where *r*, *g* and *b* are obtained from *R*, *G* and *B* by means of the standard *RGB* values, *RsRGB*, *GsRGB* and *BsRGB*, respectively. Since the formulas for each one of the components are similar, here, we will only show those equations associated with component *r*:

$$r = \begin{cases} \frac{RsRGB}{12.92}, & \text{if } RsRGB \leqslant 0.03928 \\ \left(\frac{RsRGB+0.055}{1.055}\right)^{2.4}, & \text{otherwise} \end{cases} \tag{3}$$

and *RsRGB* is defined as:

$$RsRGB = \frac{R}{255} \tag{4}$$

Calculating the contrast ratio requires obtaining the relative luminance of two colors, that is, l_1 and l_2, by means of the previous formulas. The result of the calculation of the contrast ratio *cr*, with the following table values, can be seen in Table 1:

$$cr = \frac{l_1}{l_2} \tag{5}$$

Once the combinations considered as high contrast have been obtained, we can concentrate on finding the background and character colors. To calculate the background and character colors of a bitmap, it must be scrolled pixel by pixel. The color for each bitmap pixel is obtained by associating it with one of the high contrast colors obtained. Said association consists of assigning the color obtained to the most similar color from among the pre-defined ones.

$$B_{color}[x, y] = RGB[z] \tag{6}$$

where $RGB[z]$ is the high contrast color that minimizes the following sum of partial differences:

Table 1
Result of the application of the contrast ratio *cr*. The numbers in bold represent high contrast values (above a threshold of 5).

BG\FG	Black	Blue	Green	Cyan	Red	Magenta	Brown	White	Grey	LightBlue	LightGreen	LightCyan	LightRed	Pink	Yellow	LightWhite
Black	1	1.3	4.0	4.3	1.9	2.2	**5.0**	**11.5**	**5.3**	2.4	**15.3**	**16.7**	**5.2**	**6.6**	**19.5**	**21.0**
Blue	1.3	1	3.1	3.3	1.4	1.6	3.8	**8.8**	4.0	1.8	**11.6**	**12.7**	4.0	**5.1**	**14.9**	**16.0**
Green	4.0	3.0	1	1.0	2.1	1.8	1.2	2.8	1.3	1.6	3.7	4.0	1.2	1.6	4.7	**5.1**
Cyan	4.3	3.3	1.0	1	2.2	1.9	1.1	2.6	1.2	1.7	3.4	3.8	1.1	1.5	4.4	4.7
Red	1.9	1.4	2.1	2.2	1	1.1	2.6	**6.0**	2.7	1.2	**7.9**	**8.7**	2.7	3.4	**10.1**	**10.9**
Magenta	2.2	1.6	1.8	1.9	1.1	1	2.2	**5.1**	2.3	1.1	**6.8**	**7.5**	2.3	3.0	**8.8**	**9.4**
Brown	**5.0**	3.8	1.2	1.1	2.6	2.2	1	2.3	1.0	2.0	3.0	3.3	1.0	1.3	3.9	4.1
White	**11.5**	**8.8**	2.8	2.6	**6.0**	**5.1**	2.3	1	2.1	4.7	1.3	1.4	2.1	1.7	1.6	1.8
Grey	**5.3**	4.0	1.3	1.2	2.7	2.3	1.0	2.1	1	2.1	2.8	3.1	1.0	1.2	3.6	3.9
LightBlue	2.4	1.8	1.6	1.7	1.2	1.1	2.0	4.7	2.1	1	**6.2**	**6.8**	2.1	2.7	**8.0**	**8.5**
LightGreen	**15.3**	**11.6**	3.7	3.4	**7.9**	**6.8**	3.0	1.3	2.8	**6.2**	1	1.0	2.9	2.2	1.2	1.3
LightCyan	**16.7**	**12.7**	4.0	3.8	**8.7**	**7.5**	3.3	1.4	3.1	**6.8**	1.0	1	3.1	2.5	1.1	1.2
LightRed	**5.2**	4.0	1.2	1.1	2.7	2.3	1.0	2.1	1.0	2.9	3.1	1	1.2	3.7	3.9	2.4
Pink	**6.6**	**5.1**	1.6	1.5	3.4	3.0	1.3	1.7	1.2	2.7	2.2	2.5	1.2	1	2.9	3.1
Yellow	**19.5**	**14.9**	4.7	4.4	**10.1**	**8.8**	3.9	1.6	3.6	**8.0**	1.2	1.1	3.7	2.9	1	1.0
LightWhite	**21.0**	**16.0**	**5.1**	4.7	**10.9**	**9.4**	4.1	1.8	3.9	**8.5**	1.3	1.2	3.9	3.1	1.0	1

$$|B_R[x,y] - R[z]| + |B_G[x,y] - G[z]| + |B_B[x,y] - B[z]| \quad (7)$$

Notice that $B_R[x,y]$, $B_R[x,y]$ and $B_R[x,y]$ are the red, green and blue components of pixel $B_{RGB}[x,y]$, respectively, and $R[z]$, $G[z]$ and $B[z]$ are the red, green and blue components of the high contrast color $RGB[z]$.

Therefore, in $B_{color}[x,y]$ we have one of the 16 possible basic colors selected in the previous step.

5.2. Assigning high contrast color to bitmap background and foreground

The background color assigned to the bitmap is the most highly present color in the bitmap within the palette of predetermined basic colors. Therefore:

$$bg_{RGB}(i,j) = RGB[z] \quad (8)$$

where z is obtained as the index which maximizes the following double sum:

$$N[z] = \sum_{x=1}^{w} \sum_{y=1}^{h} 1, \text{if} B_{color}[x,y] = RGB[z] \quad (9)$$

As for the foreground color of the bitmap, it is obtained as the color in order of decreasing number of occurrences which meets the criterion of high contrast with regard to the background color. It will also belong to the basic color palette.

$$fg_{RGB}(i,j) = RBG[z] \quad (10)$$

where z is the first index in decreasing order of occurrences, $N[z]$, and which also fulfills the following equation (pertaining to the contrast ratio):

$$\frac{L[z]}{L_{bg}} > 5 \quad (11)$$

$L[z]$ is the relative luminance of the predefined color $RGB[z]$ and L_{bg} is the relative luminance of the background color previously obtained fg_{RGB}.

6. Learning the thresholds

This section will describe how we addressed the learning of threshold value, $\theta_{opt}(i,j)$, which separates the foreground from the background to improve the success rate in the recognition of characters captured from a screen by a color camera. The specifications from the MIL library literally say "the target image should have a clearly-defined threshold between the characters and their background. Note that the threshold must preserve the shape of the characters. Broken and/or touching characters can degrade the results". This reaffirms the need to learn what the best threshold is for efficient segmentation.

A diagram of the proposed solution can be seen in Fig. 5. As seen in said figure, from a colored-captured image with three components (R, G, B)(*RGB* color space), we propose converting it to another color space, specifically the *HLS* (hue–lightness–saturation) space, considered more appropriate for segmentation, and selecting the best component H or L in this space. This function is called *selecting* lightness or hue component. It has also been discovered that in this new color space *HLS*, it is enough to work with only one of its three components. Once we have selected the component to work with, we propose applying filters (function *enhancing* the bitmaps) to improve the image. Later, the image is thresholded with different threshold values, θ_{trial}, through the function *thresholding* with the purpose selecting the threshold with the lowest failure rate during recognition.

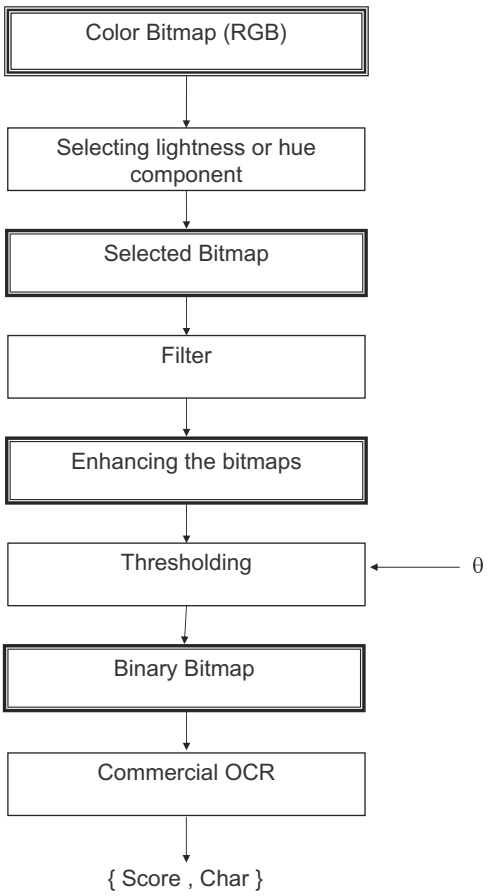

Fig. 5. Diagram of the recognition system.

The output for the proposal will be a double one. We will obtain (1) the code for the recognized character, *char*(i,j), and (2) the score value, *score*(i,j), gotten in the recognition of each character. These values will be directly related to the threshold value used to binarize the cell for each one of the characters correctly recognized.

Next, we will discuss in detail the different modules briefly described previously and which shape the whole learning system for recognition.

6.1. Selecting lightness or hue component

This block is in charge of selecting a certain component on which to work. The most detailed implementation in this block can be seen in Fig. 6. Function *transforming* from RGB to HLS color space changes the bitmaps from the *RGB* color space to *HLS*. Once the *HLS* components are obtained, the following elements from the module (*calculating* lightness and hue foreground mean value and *calculating* lightness and hue background mean value) have the goal of selecting only one of these components as the input to the following module. The goal is to choose the components which will subsequently enable a better performance in character and background segmentation (through thresholding). Experimental results show that it is possible to ignore the component saturation (component S), thus only making it necessary to calculate which of the two components left (H or L) is the most appropriate for subsequent segmentation (thresholding).

6.1.1. Transforming from RGB to HLS color space

Through *transforming* from RGB to HLS color space, the image captured in color space *RGB* is converted into a more appropriate

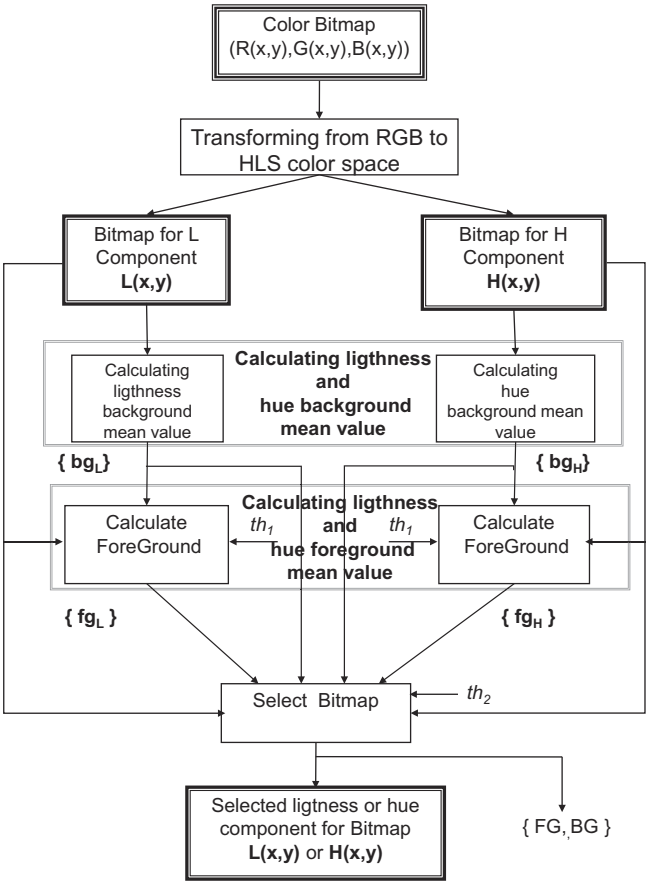

Fig. 6. Diagram of the selection component system.

color space with the purpose of carrying out character segmentation on the background. The color space chosen is the *HLS* space (Smith, 1978), where the information on luminosity (component L) is clearly separate from the information associated with the color (components H and S).

6.1.2. Calculating lightness and hue background mean value

This element obtains the background luminosity value, $\overline{bg_L}$, and the background hue value, $\overline{bg_H}$. This value corresponds to the mean value in each cell border. Using the first two rows, the last two rows, the first two columns and the last two columns of pixels in a cell, as representative of the cell border is considered to be enough. If we will recall that a cell (i,j) has a width of w pixels times a height of h pixels, then the total number of pixels in the border is $(4 \times (w+h)) - 16$.

Therefore, the background value for component H, $\overline{bg_H}$, comes from Eq. (12) and the background value for component L, $\overline{bg_L}$, through Eq. (13), which gather the mean values $L[x,y]$ and $H[x,y]$, respectively, for pixels $[x,y]$, which belong to a given cell $B_{H/L}(i,j)$.

$$\overline{bg_H} = \frac{\sum_{[x,y]\in border} H[x,y]}{(4 \times (w+h)) - 16} \tag{12}$$

$$\overline{bg_L} = \frac{\sum_{[x,y]\in border} L[x,y]}{(4 \times (w+h)) - 16} \tag{13}$$

where

$$border = \{[x,y] | x \in \{0,1,w-1,w\}, y \in \{0,1,h-1,h\}\} \tag{14}$$

6.1.3. Calculating lightness and hue foreground mean value

This function, on the other hand, calculates the foreground value for components L and H, $\overline{fg_L}$ and $\overline{fg_H}$, respectively. The value of $\overline{fg_L}$ corresponds to the mean of values $L[x,y]$ which do not belong to the border of the bitmap and differ in more than one amount, Δ_{border}, from the background value obtained. Similarly, the value for $\overline{fg_H}$ is assigned.

Eqs. (15), (16) and (19) show the calculation of this mean for component L, $\overline{fg_L}$, and Eqs. (17), (18) and (20) for component H, $\overline{fg_H}$, respectively.

In the case of component L, for each pixel $[x,y]$, Eq. (15) associates a value different from 0 to those pixels whose lightness $L[x,y]$ is equal to or greater than Δ_{border}. That is, we keep pixel lightness $fg_{L[x,y]} = L[x,y]$. On the other hand, Eq. (16) enables us to obtain $t_{L[x,y]}$ where value 1 is stored for those pixels whose $L[x,y]$ is equal to or greater than Δ_{border}, and 0, otherwise. This variable $t_{L[x,y]}$ will enable us to count how many pixels belong to the character, dismissing those which are part of the background of the bitmap.

$$fg_{L[x,y]} = \begin{cases} L[x,y], & \text{if } |L[x,y] - \overline{bg_L}| \geqslant \Delta_{border} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

$$t_{L[x,y]} = \begin{cases} 1, & \text{if } |L[x,y] - \overline{bg_L}| \geqslant \Delta_{border} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

Eqs. (17) and (18) operate in the same way, but for component H.

$$fg_{H[x,y]} = \begin{cases} H[x,y], & \text{if } |H[x,y] - \overline{bg_H}| \geqslant \Delta_{border} \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

$$t_{H[x,y]} = \begin{cases} 1, & \text{if } |H[x,y] - \overline{bg_H}| \geqslant \Delta_{border} \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

This way, from the values obtained, $fg_{L[x,y]}$ and $t_{L[x,y]}$, and through Eq. (19), or from the values obtained, $fg_{H[x,y]}$ and $t_{H[x,y]}$, and through Eq. (20), we obtain $\overline{fg_L}$ and $\overline{fg_H}$ as the means for values $L[x,y]$ and $H[x,y]$ and which do not belong to the border of the bitmap and differ in more than one amount, Δ_{border}, from the background value obtained.

$$\overline{fg_L} = \frac{\sum_{[x,y]\notin border} L[x,y]}{\sum_{[x,y]\notin border} t_{L[x,y]}} \tag{19}$$

$$\overline{fg_H} = \frac{\sum_{[x,y]\notin border} H[x,y]}{\sum_{[x,y]\notin border} t_{H[x,y]}} \tag{20}$$

6.1.4. Selecting lightness or hue component for bitmap

Once values H and L for the background $(\overline{bg_H}, \overline{bg_L})$ as well as the character $(\overline{fg_H}, \overline{fg_L})$ are obtained, we select which of the two components will be used for subsequent recognition through the commercial OCR, $B_{H/L}(i,j)$.

Through experience, we have come to the conclusion that it is better to work with component L, except in cases where the difference in luminosity between the background and the character is less than a certain amount, called $\Delta_{bg/fg}$. In that case, we work with the hue component H.

$$B_{H/L}(i,j) = \begin{cases} B_L(i,j), & \text{if } |\overline{fg_L}(i,j) - \overline{bg_L}(i,j)| \geqslant \Delta_{bg/fg} \\ B_H(i,j), & \text{otherwise} \end{cases} \tag{21}$$

Finally, the background and foreground colors of the bitmap are assigned to the mean values previously calculated, according to the color component selected:

$$FG(i,j) = \begin{cases} \overline{fg_L}(i,j), & \text{if } |\overline{fg_L}(i,j) - \overline{bg_L}(i,j)| \geqslant \Delta_{bg/fg} \\ \overline{fg_H}(i,j), & \text{otherwise} \end{cases} \tag{22}$$

$$BG(i,j) = \begin{cases} \overline{bg_L}(i,j), & \text{if } |\overline{fg_L}(i,j) - \overline{bg_L}(i,j)| \geqslant \Delta_{bg/fg} \\ \overline{bg_H}(i,j), & \text{otherwise} \end{cases} \tag{23}$$

6.2. Enhancing the bitmaps

In this part of the process, different types of filters are applied to each bitmap $B_{H/L}(i,j)$. The goal of this successive filtering is to improve the bitmap so it can be analyzed in the most optimal way possible by module OCR. Next, we show the different processes carried out on each bitmap in the order applied.

First, a 5×5 "enhancement" mask is applied to each bitmap, $B_{H/L}$,as shown in Eq. (24), to obtain the background characters, B_f, in a more distinguishable manner.

$$B_f[x,y] = B_{H/L}[x,y] \circ \begin{vmatrix} 1 & -2 & 3 & -2 & 1 \\ -2 & 3 & 5 & 3 & -2 \\ 3 & 5 & 9 & 5 & 3 \\ -2 & 3 & 5 & 3 & -2 \\ 1 & -2 & 3 & -2 & 1 \end{vmatrix} \tag{24}$$

Next, a 2×2 erosion filter, as shown in Eq. (25) is applied, to limit the thickness of the character. The previously applied 5×5 enhancement filter unfortunately introduces an undesired effect of blurring the character borders. This effect is now corrected by means of the erosion filter, obtaining a better defined shape.

$$B_e[x,y] = \min_{[x',y'] \in [0..1,0..1]} B_f(x+x', y+y') \tag{25}$$

6.3. Thresholding

Once the component (L or H) we are going to work with is selected, and once the quality of the bitmap $B_{H/L}(i,j)$ is improved, the next milestone is to select the optimal threshold to distinguish between the background and the character. To do so, an initial threshold, θ_{init}, is established in the first place. The initial binarization threshold corresponds to the mean between the foreground FG value obtained and the background BG, as shown in Eq. (26).

$$\theta_{init} = \frac{FG + BG}{2} \tag{26}$$

Due to the problems derived from the noise always present in images, this threshold value is not always the most appropriate. The threshold value depends on the position of the character on the screen, as well as the morphology of the character. In fact, the threshold depends on the position of the character on the screen, since the conditions for luminosity do not have to be homogeneous in the whole screen. Likewise, experimental results show that larger or more closed characters need a different threshold value from smaller characters.

Therefore, we propose increasing to or decreasing from the value of θ_{init} an amount Δ_θ a number of times between κ_{min} and κ_{max}. Whether to increase or decrease from the value of θ_{init} depends on the relation between background color and foreground color, as shown in Eq. (27).

$$\theta_{trial} = \begin{cases} \theta_{init} - \kappa \times \Delta_\theta, & \text{if } FG < BG \\ \theta_{init} + \kappa \times \Delta_\theta, & \text{otherwise} \end{cases} \tag{27}$$

where $\kappa \in [\kappa_{min}..\kappa_{max}]$.

The commercial OCR used for recognition uses as input the binarized bitmap $B_b(i,j)$ with a black character and a white background. If $FG < BG$, the binarization given by Eq. (28) will be applied, and if $FG \geqslant BG$, the binarization given by formula (29) will be applied.

$$B_b(i,j) = \begin{cases} 0, & \text{if } B_e(i,j) \leqslant \theta_{trial} \\ 255, & \text{otherwise} \end{cases} \tag{28}$$

$$B_b(i,j) = \begin{cases} 0, & \text{if } B_e(i,j) \geqslant \theta_{trial} \\ 255, & \text{otherwise} \end{cases} \tag{29}$$

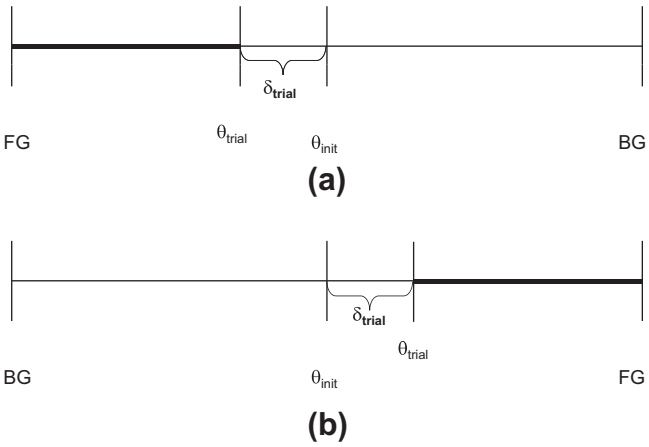

Fig. 7. Diagram of both binarization scenarios. (a) The background value is greater than the character value. (b) The character value is greater than the background value.

Defining $\theta_{trial} = \kappa \times \Delta_\theta$, Fig. 7 shows, schematically, the two different binarizations depending on the values obtained for FG and BG. Fig. 7(a) represents the binarization when the background value, BG, is greater than the character value, FG. In this case, the thickest part of the segment, $[FG \leftrightarrow \theta_{trial}]$, represents the value range (luminosity or hue) which will be associated with the character after binarization. Fig. 7(a) represents binarization when the background value, BG, is lower than the character value, FG. In this case, the thickest part of the segment, $[\theta_{trial} \leftrightarrow \theta_{trial}FG]$, represents the value range associated with the character after binarization. Notice that in both cases, δ_{trial} is the displacement for value θ_{init} previously calculated through Eq. 26.

Let variables δ^{hit}_{min} and δ^{hit}_{max} be two values associated with those minimum and maximum thresholds that yield accurate recognition. We have:

$$\delta^{hit}_{min} = \kappa^{hit}_{min} \times \Delta_\theta \tag{30}$$

$$\delta^{hit}_{max} = \kappa^{hit}_{max} \times \Delta_\theta \tag{31}$$

For each character and for each possible bitmap position in the display, the values of these variables δ^{hit}_{min} and δ^{hit}_{max} are stored in a matrix.

Fig. 8 shows the variation of δ^{hit}_{min} for position $(1,1)$ for the different ASCII codes of the character used for training the recognition system. The graph shows that although there are some values for $\delta^{hit}_{min} = 0$, most of them need values greater than δ^{hit}_{min} to yield accurate hits. Also notice that the maximum value of δ^{hit}_{min} is 40.

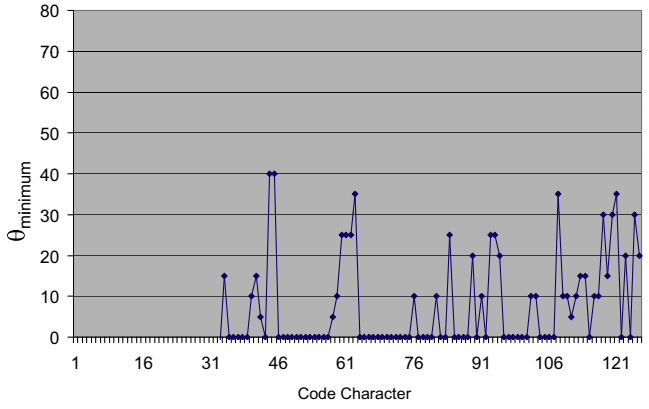

Fig. 8. Variation of δ^{hit}_{min}, depending on the character, for position $(1,1)$.

Similarly, Fig. 9 shows the values of δ^{hit}_{max} for the same position and the same training set. Once again, we notice that obviously, the values will be equal or greater. If they match, it will indicate that there is only one possible value of δ_{trial} which will yield an accurate hit for that character.

Fig. 10 shows the variation of δ^{hit}_{min} for a specific ASCII character (character *A*), according to its position of the bitmap associated with the character. The input image to the recognition system has 493 cells distributed along 29 columns and 17 rows. Specifically, the position is $(row \times 29) + column$. Notice that in Fig. 10, the main variations occur when changing rows (values which are multiples of 29).The reason for this is that they are the most distant positions in axis *x*.

As discussed earlier, for each character used in training, a matrix or table is generated for the values of δ^{hit}_{min} and δ^{hit}_{max} in each position of the display. Table 2 shows the values for $[\delta^{hit}_{min} \leftrightarrow \delta^{hit}_{max}]$ gotten for a portion of an 8×8 cell image for a specific character (character *A*).

6.4. Calculating the segmentation score range

The commercial OCR used provides the code for the recognized character, as well as a score value, *score*(*i*,*j*). For each character, a matrix with values $score_{min}$ and $score_{max}$ is stored for each position. This score value is stored to be used later on in the detection phase.

7. Using the learned parameters for optical recognition

As discussed earlier, the training phase proposed in this article aims to choose the threshold with the highest accuracy rate in recognition. This threshold will be called optimal, θ_{opt}. This section will explain how to determine the value associated with each position of the character to be recognized.

To get this optimal value, the recognition system will have been trained with different screens, each having the same character throughout the whole screen. From the values for δ^{hit}_{min} and δ^{hit}_{max} gotten for all training characters (and all screens), the value of θ_{opt} with the highest number of occurrences in all positions throughout the training will be chosen. Therefore, after training the system for all possible characters in every possible position of the display, we have matrices containing $[\delta^{hit}_{min}(i,j,ch) \leftrightarrow \delta^{hit}_{max}(i,j,ch)]$.

$$\S(i,j,\delta) = \sum_{ch} 1, \text{if } \delta^{hit}_{min}(i,j,ch) \leqslant \delta \leqslant \delta^{hit}_{max}(i,j,ch) \tag{32}$$

$$\delta_{opt}(i,j) = argmax_{\delta}\S(i,j,\delta) \tag{33}$$

$\kappa_{min} \times \Delta_{\theta} \leqslant \delta \leqslant \kappa_{max} \times \Delta_{\theta}$.

For each threshold value gotten, the number of occurrences is obtained by counting the number of times it appears at each interval.

$$\theta_{opt}(i,j) = \begin{cases} \theta_{init} + \delta_{opt}(i,j), & \text{if } FG < BG \\ \theta_{init} - \delta_{opt}(i,j), & \text{otherwise} \end{cases} \tag{34}$$

For example, if the results are that for a certain position of a certain character, the interval for δ^{hit}_{min} and δ^{hit}_{max} is $[0 \leftrightarrow 20]$, the value of the occurrences for all interval values will increase by 1. In this example in particular, using a value of $\Delta_{\theta} = 5$ as the interval increment, the occurrences of values 0,5, 10, 15 and 20 will increase.

Fig. 11 shows the histogram for the number of occurrences for a certain position (position (1,1)) throughout the whole training.

Thus, for each position (*i*,*j*) in the image, we obtain a histogram. Those values δ_{opt} which provide the greatest number of occurrences are stored in a matrix. Table 3 shows the values obtained for δ_{opt} in the different positions during the training. As shown, the value for position (1,1) in Table 3 coincides with the value for δ_{opt} provided for the histogram's maximum value shown in Fig. 11.

The value for θ_{opt} generated from the value for δ_{opt} which has generated the highest number of occurrences, along with the table or matrix obtained for $score_{min}$ and $score_{max}$, are later used for recognition in the following way:

1. A character in position (*i*,*j*) is read using the value for θ_{opt} obtained during the training phase for the position (*i*,*j*) indicated.
2. The value for θ_{opt} is checked to see that it is in the range associated for the same character in the training phase, $\delta^{hit}_{min} \leftrightarrow \delta^{hit}_{max}$.

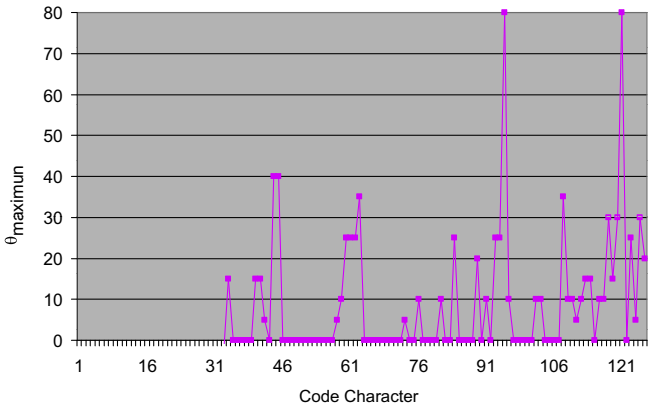

Fig. 9. Variation of δ^{hit}_{max}, depending on the character, for position (1,1).

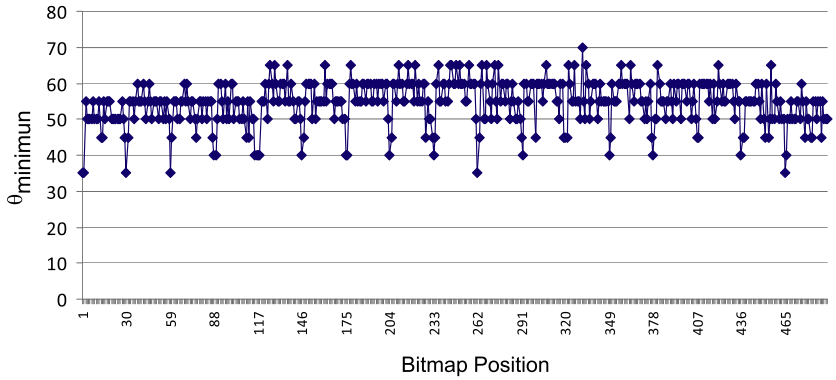

Fig. 10. Variation of δ^{hit}_{min} for ASCII character *A*.

Table 2
Table of values δ_{min}^{hit} and δ_{max}^{hit} for a sub-image of size 8×8.

[0 ↔ 60]	[0 ↔ 45]	[0 ↔ 10]	[0 ↔ 45]	[0 ↔ 50]	[0 ↔ 50]	[0 ↔ 15]	[0 ↔ 10]
[0 ↔ 30]	[0 ↔ 50]	[0 ↔ 15]	[5 ↔ 15]	[0 ↔ 30]	[0 ↔ 55]	[0 ↔ 15]	[0 ↔ 55]
[0 ↔ 65]	[0 ↔ 65]	[0 ↔ 30]	[0 ↔ 0]	[0 ↔ 60]	[0 ↔ 30]	[0 ↔ 35]	[20 ↔ 20]
[0 ↔ 60]	[0 ↔ 30]	[0 ↔ 20]	[30 ↔ 30]	[0 ↔ 30]	[0 ↔ 55]	[0 ↔ 65]	[20 ↔ 65]
[0 ↔ 50]	[0 ↔ 55]	[0 ↔ 65]	[0 ↔ 0]	[0 ↔ 70]	[0 ↔ 80]	[0 ↔ 0]	[30 ↔ 70]
[0 ↔ 70]	[0 ↔ 25]	[5 ↔ 15]	[10 ↔ 35]	[15 ↔ 55]	[0 ↔ 25]	[50 ↔ 60]	[25 ↔ 30]
[0 ↔ 60]	[10 ↔ 45]	[35 ↔ 40]	[20 ↔ 20]	[20 ↔ 35]	[15 ↔ 15]	[35 ↔ 60]	[0 ↔ 45]
[10 ↔ 60]	[25 ↔ 60]	[25 ↔ 65]	[10 ↔ 35]	[15 ↔ 70]	[15 ↔ 75]	[20 ↔ 50]	[0 ↔ 45]

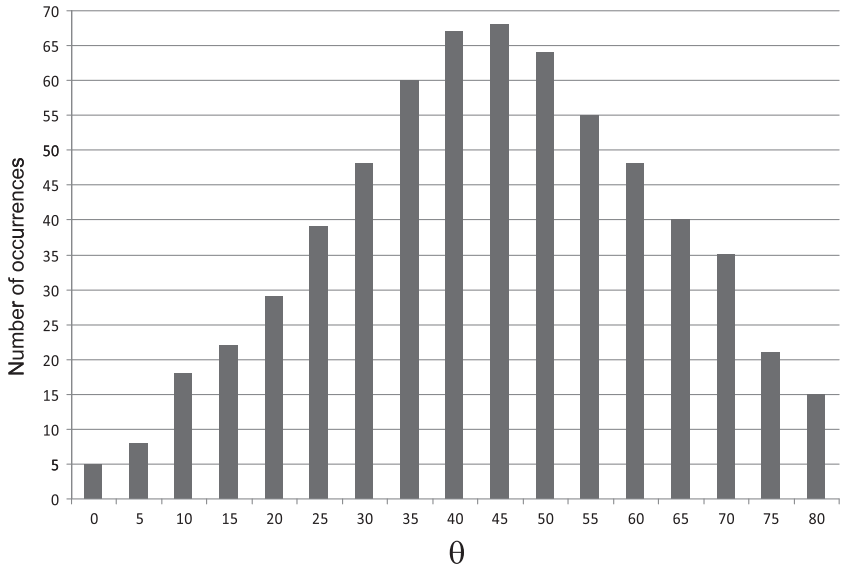

Fig. 11. Histogram for threshold occurrences for position (1,1) throughout the whole training.

Table 3
Values obtained for δ_{opt} during training.

35 35 55 50 50 50 50 55 50 50 50 55 45 45 55 50 55 55 55 50 50 50 50 50 50 50 55 50 45
35 45 55 55 55 50 55 60 55 55 55 60 55 50 55 60 55 50 55 55 55 50 55 55 50 55 50 55 50
35 45 55 55 55 50 50 55 55 60 55 60 55 50 55 55 50 45 50 55 55 50 55 55 50 55 55 55 45
40 40 50 60 60 55 50 60 50 55 50 60 60 50 55 55 55 50 50 55 50 45 55 45 55 50 50 40 40
40 40 55 55 55 60 50 60 65 60 55 65 60 55 55 60 60 60 55 65 55 55 60 55 50 50 55 55 50
40 45 55 60 60 60 50 60 50 55 55 55 55 55 65 55 60 60 60 60 50 55 55 55 55 50 50
40 40 60 65 60 60 55 60 55 55 55 60 60 55 60 60 60 55 55 60 60 55 60 60 60 55 60 60 50
40 45 60 60 55 60 65 60 60 55 55 60 65 60 60 60 55 65 55 60 55 55 60 60 45 55 50 50 45
40 45 60 65 55 55 60 60 55 55 60 65 65 60 60 65 60 65 60 60 60 55 55 65 60 60 60 60 50
35 45 60 65 50 50 65 60 55 50 60 65 55 50 65 55 60 60 55 60 60 50 55 60 55 50 50 55 50
45 40 60 60 55 55 60 60 60 45 60 60 60 60 55 60 65 60 60 60 60 60 60 55 55 50 60 60 45
45 45 65 60 55 55 65 55 55 55 55 50 70 55 50 65 60 50 55 60 60 60 50 55 60 55 55 55 55 55
40 45 60 55 55 55 60 60 65 60 60 60 60 50 65 60 55 60 60 60 60 55 55 50 55 50 55 60 45
40 50 50 65 60 55 55 55 50 55 55 60 55 50 60 55 60 60 50 60 60 60 55 60 60 50 55 60 50
45 45 60 60 60 60 60 60 55 60 50 60 50 55 65 60 55 60 55 55 60 60 60 60 55 50 60 55 55
40 45 45 55 55 55 55 55 55 55 55 60 60 55 50 60 50 45 60 45 50 65 50 50 60 55 50 55 50 50
35 40 50 50 55 50 50 50 55 55 50 60 55 45 55 50 50 45 45 55 55 50 55 55 45 55 50 50 50

Table 4
Hit percentage for all ASCII characters.

Char code	% Hits	Char code	% Hits	Char code	% Hits	Char code	% Hits
33	10	57	94	81	35	105	99
34	100	58	100	82	77	106	81
35	100	59	100	83	100	107	100
36	100	60	100	84	71	108	86
37	95	61	100	85	52	109	87
38	84	62	100	86	99	110	99
39	100	63	13	87	99	111	83
40	100	64	67	88	100	112	100
41	100	65	94	89	100	113	100
42	100	66	86	90	78	114	100
43	100	67	53	91	77	115	84
44	89	68	67	92	100	116	100
45	100	69	40	93	60	117	87
46	100	70	73	94	99	118	100
47	100	71	71	95	81	119	92
48	92	72	98	96	100	120	99
49	100	73	68	97	90	121	99
50	73	74	69	98	86	122	89
51	95	75	99	99	88	123	100
52	100	76	66	100	88	124	100
53	76	77	98	101	83	125	94
54	83	78	95	102	98	126	97
55	83	79	30	103	94		
56	52	80	78	104	100		

3. If it is within that range and has a score $score_{read}$ between the minimum score $score_{min}$ and the maximum score $score_{max}$ for that character, it is considered accurate. Otherwise, the OCR reading will be done again, but using a threshold within the range associated for the character read.

8. Data and results

This section shows the results obtained for recognition. To obtain these results, the values have been set as follows: $\kappa_{min} = 0$, $\kappa_{max} = 16$ and $\Delta_0 = 5$. $\Delta_{border} = 60$ and $\Delta_{bg|fg} = 3$, for luminosity, $\Delta_{bg|fg} = 40$, for hue.

The tests performed have demonstrated the capabilities of the system in relation to the optical character recognition task. In order to get the necessary displays for performing the tests, a simulator has been developed. The simulator is generic, enabling to configure the characteristics of any kind of display, CRT, LCD, and TFT-LCD. Due to the generality of the simulator, the size of a simulated display (rows and columns) may be easily modified for generating a wide range of displays.

In this article we offer the results of testing the character segmentation on a complete set of ASCII characters (from character code 33–126). The mean results of the recognition may be observed on Table 4, where the mean hit percentage overcomes an 86%, throwing a hit of 100% for 32 different characters, and a hit greater than an 80% for 71 different characters. There are only two characters offering a very poor hit percentage, namely, ASCII characters 33 and 66, corresponding to ? and ! symbols, respectively. This is a problem of the commercial OCR, as the library handles very badly the characters that present unconnected elements (formed by more than one shape).

9. Conclusions

In this paper text segmentation in generic displays has been proposed. For this purpose, the best binarization values for a commercial OCR system are learnt. The aim of this work is to provide the capability to automatically evaluate standard textual display information, so that tasks that involve visual user verification can be performed without human intervention.

The proposed system for generic displays includes some of the usual text recognition steps, namely localization, extraction and enhancement, and optical character recognition. In textual displays the characters use to be placed at fixed positions. Therefore, our solution establishes a set of bitmaps in the display, in accordance with the number of rows and columns that the display is able to generate. The proposal has been tested on a multi-display simulator and a commercial OCR system, throwing good initial results.

Acknowledgements

This work was partially supported by Spanish Ministerio de Ciencia e Innovación TIN2010-20845-C03-01 grant.

References

Aas, K., & Eikvil, L. (1996). Text page recognition using grey-level features and hidden Markov models. *Pattern Recognition, 29*(6), 977–985.

Antani, S., Crandall, D., & Kasturi, R. (2000). Robust extraction of text in video. In *Proceedings of the 15th international conference on pattern recognition* (Vol. 1, pp. 831–834).

Shen, D., & Ip, H. H. S. (1997). A Hopfield neural network for adaptive image segmentation: An active surface paradigm. *Pattern Recognition Letters, 18*, 37–48.

Chen, D., Shearer, K., & Bourlard, H. (2001). Text enhancement with asymmetric filter for video OCR. In *Proceedings of the 11th international conference on image analysis and processing* (pp. 192–198).

Cho, S., Haralick, R., & Yi, S. (1989). Improvement of Kittler and Illingworths's minimum error thresholding. *Pattern Recognition, 22*, 609–617.

Faugeras, O. (1993). *Three-dimensional computer vision: A geometric viewpoint*. The MIT Press.

Gupta, M. R., Jacobson, N. P., & Garcia, E. K. (2007). OCR binarization and image preprocessing for searching historical documents. *Pattern Recognition, 40*, 389–397.

Hori, O. (1999). A video text extraction method for character recognition. In *Proceedings of the international conference on document analysis and recognition* (pp. 25–28).

Huang, S., Ahmadi, M., & Sid-Ahmed, M. A. (2008). A hidden Markov model-based character extraction method. *Pattern Recognition, 41*, 2890–2900.

Jain, A. K., & Bhattacharjee, S. (1992). Text segmentation using gabor filters for automatic document processing. *Machine Vision and Applications, 5*, 169–184.

Jawahar, C. V., Biswas, P. K., & Ray, A. K. (1997). Investigations on fuzzy thresholding based on fuzzy clustering. *Pattern Recognition, 30*, 1605–1613.

Jung, C., Liu, Q., & Kim, J. (2008). A new approach for text segmentation using a stroke filter. *Signal Processing, 88*, 1907–1916.

Kamada, H., & Fujimoto, K. (1999). High-speed, high-accuracy binarization method for recognizing text in images of low spatial resolutions. In *Proceedings of the international conference on document analysis and recognition* (pp. 139–142).

Kanungo, T., Haralick, R. M., Baird, H. S., Stuezle, W., & Madigan, D. (2000). A statistical, nonparametric methodology for document degradation model validation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(11), 1209–1223.

Kittler, J., & Illingworth, J. (1986). Minimum error thresholding. *Pattern Recognition, 19*, 41–47.

Kohmura, H., & Wakahara, T. (2006). Determining optimal filters for binarization of degraded characters in color using genetic algorithms. In *Proceedings of the 18th international conference on pattern recognition, ICPR 2006* (Vol. 3, pp. 661–664).

Lee, S.-W., & Kim, Y.-J. (1995). Direct extraction of topographic features for gray scale character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 17*(7), 724–729.

Li, H., Doermann, D., & Kia, O. (2000). Automatic text detection and tracking in digital video. *IEEE Transactions on Image Processing, 9*, 147–156.

Li, H., & Doermann, D. (2000). A video text detection system based on automated training. In *Proceedings of the IEEE international conference on pattern recognition* (pp. 223–226).

Likforman-Sulem, L., & Sigelle, M. (2008). Recognition of degraded characters using dynamic Bayesian networks. *Pattern Recognition, 41*, 3092–3103.

Liu, C., Wang, C., & Dai, R. (2006). Low resolution character recognition by image quality evaluation. In *Proceedings of the 18th international conference on pattern recognition* (Vol. 1, pp. 864–867).

Makridis, M., Nikolaou, N., & Papamarkos, N. (2010). An adaptive technique for global and local skew correction in color documents. *Expert Systems with Applications, 37*, 6832–6843.

Montazer, G. A., Saremi, H. Q., & Khatibi, V. (2010). A neuro-fuzzy inference engine for Farsi numeral characters recognition. *Expert Systems with Applications, 37*, 6327–6337.

Otsu, N. (1979). A threshold selection method from gray-level histogram. *IEEE Transactions on Systems, Man, and Cybernetics, 9*, 62–66.

Park, H. S., & Lee, S. (1998). Off-line recognition of large-set handwritten characters with multiple hidden Markov models. *Pattern Recognition, 31*, 1849–1864.

Pikaz, A., & Averbuch, A. (1996). Digital image thresholding based on topological stable state. *Pattern Recognition, 29*, 829–843.

Prewitt, J. M. S., & Mendelsohn, M. L. (1965). The analysis of cell images. *Annals of the New York Academy of Sciences, 128*(3), 1035–1053.

Sato, T., Kanade, T., Hughes, E. K., & Smith, M. A. (1998). Video OCR for digital news archives. In *Proceedings of the IEEE workshop on content based access of image and video databases* (pp. 52–60).

Schenkel, M., & Jabri, M. (1998). Low resolution degraded document recognition using neural networks and hidden Markov models. *Pattern Recognition Letters, 19*, 365–371.

Shapiro, V., Gluhchev, G., & Dimov, D. (2006). Towards a multinational car license plate recognition system. *Machine Vision and Applications, 17*, 173–183.

Smith, M. A., & Kanade, T. (1995). *Video skimming for quick browsing based on audio and image characterization*. Technical report CMU-CS-95-186, Computer Science Department, Carnegie Mellon University.

Tsai, R. Y. (1987). A versatile camera calibration technique for high accuracy 3-d maching vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics & Automation, 3*, 323–344.

Tsai, C. M., & Lee, H. J. (2002). Binarization of color document images via luminance and saturation color features. *IEEE Transactions on Image Processing, 11*(4), 434–451.

Taylor, M. J., & Dance, C. R. (1998). Enhancement of document images from cameras. In *Proceedings of SPIE, document recognition V* (pp. 230–241).

Trier, O. D., & Taxt, T. (1995). Evaluation of binarization methods for document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 17*, 312–315.

Smith, A. R. (1978). Color gamut transform pairs. *Computer Graphics, 12*(3), 12–19.

Wang, K., & Kangas, J. A. (2003). Character location in scene images from digital camera. *Pattern Recognition, 36*, 2287–2299.

Wang, L., & Pavlidis, T. (1993). Direct gray-scale extraction of features for character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 15*(10), 1053–1066.

<http://www.w3.org/TR/2008/REC-WCAG20-20081211/>.

Wu, V., Manmatha, R., & Riseman, E. M. (1997). Finding text in images. In *Proceedings of ACM international conference on digital libraries* (pp. 23–26).

Yan, F., Zhang, H., & Kube, C. R. (2005). A multistage adaptive thresholding method. *Pattern Recognition Letters, 26*, 1183–1191.

Yan, H., & Wu, J. (1994). Character and line extraction from color map images using a multi-layer neural network. *Pattern Recognition Letters, 15*, 97–103.

Yanni, M. K. & Horne, E. (1994). A new approach to dynamic thresholding. In *Proceedings of the seventh European signal processing conference, EUSIPCO-94* (Vol. 1, pp. 34–44).

Yanowitz, S. D., & Bruckstein, A. M. (1989). A new method for image segmentation. *Computer Vision, Graphics, and Image Processing, 46*(1), 82–95.

Zheng, Y., Li, H., & Doermann, D. (2004). Machine printed text and handwriting identification in noisy document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26*(3), 337–353.