The importance of segmentation applied to CAPTCHA recognition

Vinícius Mauricio de Almeida
Federal University of Ouro Preto (UFOP)
Department of Computer Sciences (DECOM)
viniciusmdea@gmail.com

Vinícius A. P. Queiroz
Federal University of Ouro Preto (UFOP)
Department of Computer Sciences (DECOM)
vini.apq@gmail.com

Abstract

In this work, we present a multi-binarization method for CAPTCHA segmentation, and a comparison with the results obtained by a simple hand-crafted segmentation algorithm.

We also utilize Artificial Neural Networks fed with Histogram Oriented Gradients features to train the classifier.

Our end-to-end system with the multi-binarization segmentation presents an Accuracy of 3.15% on a CAPTCHA database. The hand-crafted segmentation algorithm presents a 59.35% accuracy on the same database.

Herein we present the results from the combinations of binarization techniques and the most promising pre-processing techniques to help segregate background from foreground.

1. Introduction

Completely Automated Public Turing tests to tell Computers and Humans Apart, (or, as they are more widely known, CAPTCHAs) are a standard security techonology in various websites and softwares [1]. Nowadays, there are several methods for generating text-based CAPTCHAs, and the approach that has proved to be the most effective against anti-CAPTCHAs softwares is to rely on segmentation resistance [8]. Therefore, it seems reasonable to test the efficiency of a given CAPTCHA generator by testing several segmentation methods.

In this work, we test methods for CAPTCHA recognition. For this purpose, we divide the problem of in two sub-problems: Segmentation of the image; and Character Recognition. We experimented two methods for Image Segmentation. The first one has been proved to obtain state-of-the-art results for License Plate Segmentation [9]. The second one is a simple hand-crafted segmentation algorithm. Both methods were tested with the same Character Classifier, based on Artificial Neural Networks and HoG.

2. Related Work

Text recognition is a well-known problem, and was addressed many times in the literature, with different applications, as license plate recognition [9], house numbers identification [3], document scans [2], and many others. There are also lots of works that have their primary attention on CAPTCHA recognition, as in [1, 8].

This article differs from the previous ones as it uses a method that was proposed to be for general purpose [9] and, as we will show, is not well suited for the CAPTCHA segmentation problem. Because of that, we present a method that suits better for this purpose.

3. Technical Background

This section is meant to provide the technical background on some of the concepts utilized in this work.

3.1. Segmentation

Image Segmentation is a process of division of a given image as it suits better for its purpose. For the CAPTCHA Recognition problem, it consists on retrieving pieces of images in which there is only one character. For this purpose, it's necessary to find the positions of the divisions being applied to the image, known as the Bounding Boxes. An important factor for a well applied segmentation is the knowledge of the images' nature, as a common distribution of the objects, or the presence of different objects that are connected. With this knowledge, it is possible to define the best segmentation method. The most popular segmentation method for text recognition is known as Connected Component Analysis, and it relies on the binarization of the image.

3.2. Binarization

For a given image, it is considered to be in the binary form if it agrees with the following condition:

$$P(i,j) = \begin{cases} 1 & \text{if } \textit{foreground} \\ 0 & \text{if } \textit{background} \end{cases} \tag{1}$$

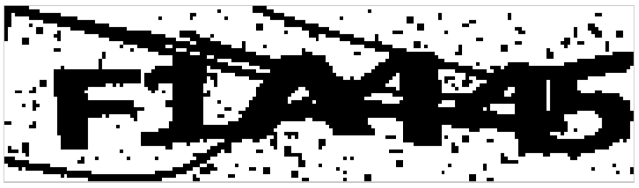
Figure 1. Example of a binarized CAPTCHA by Otsu's method

where $P(i,j)$ is the boolean intensity value of the pixel localized in the i-th row and j-th column and, concretely, *foreground* would correspond to the objects which we are interested in and *background* everything else. Note that 1 corresponds to a white pixel, and 0 to a black pixel.

Binarization algorithms consists on a sequence of commands that, given a grayscale or colored image, outputs a binary image, where the *foreground* pixels are set to 1 and *background* pixels are set to 0. There are several algorithms that does what is described here, and each one has it's advantages. These algorithms can be divided in two main branches: Local and Global.

The most common global methods rely on the histogram of the image, and usually, does not perform well when uneven lighting is present [9]. As CAPTCHAs usually does not have this issue, we test Otsu's method in our algorithm and it performs quite well, as we show in Section …

Local methods have better performances when the images presents noise and/or uneven lighting conditions. But it has the disadvantage of the necessity of choosing the parameters. A poor choice of parameters for local binarization algorithms can result in very poor binarization, and thus, bad segmentation [9].

3.2.1 Otsu

Otsu's [5] is a binarization global method, which chooses an optimal threshold by minimizing the intraclass variance. It presents a good performance, mainly when compared with other global methods. In [9], a comparison was made between different binarization methods for license plates, and Otsu's shows the best performance between global methods.

3.2.2 Niblack

Niblack proposed a local binarization [4], which calculates the pixel value, T, through the means, m, and standard deviation, s, of neighboring pixels, in a square region of width N, as this formula goes:

$$T = m + ks \tag{2}$$

Niblack had the better accuracy between the local binarization methods to segmentate license plates [9].

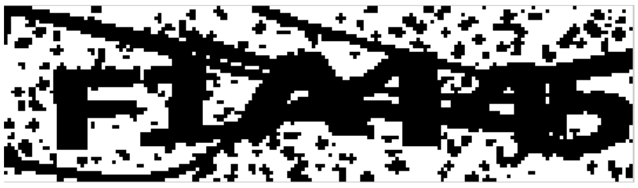
Figure 2. Example of a binarized CAPTCHA by Niblack's method

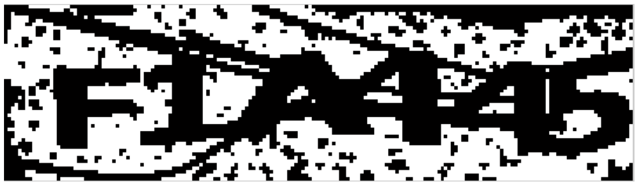
Figure 3. Example of a binarized CAPTCHA by Sauvola's method

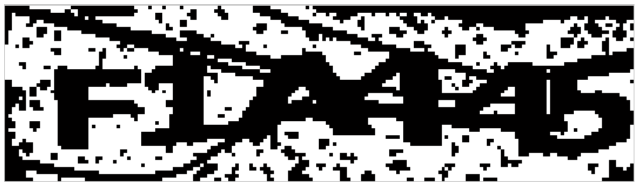
Figure 4. Example of a binarized CAPTCHA by Wolf's method

3.2.3 Sauvola and Wolf

Both Sauvola and Wolf are other local binarization methods derived from Niblack's method, with a proposed enhancement in images with backgrounds in the presence of noise and uneven lighting.

In Sauvola [6], the standard deviation s of the analised region wrapped by a square matrix of width N is divided by the dynamic of the standard deviation of the neighbouring pixels, and then decremented one unit, following this formula:

$$T = m \cdot [1 + k(s/R - 1)] \tag{3}$$

Wolf's proposed algorithm [7] modificates Sauvola's formula, in the attempt of improving the binarization in images with thin edges. In this algorithm, we have the multiplication of the second term of the formula by the difference between the lowest pixel intensity M and the mean, m, of the bounded region, as this formula:

$$T = m + k \cdot (s/R - 1) \cdot (m - M) \tag{4}$$

3.3. Connected Component Analysis

The Connected Component Analysis (CCA) is a method widely used to count objects, and is also utilized to segmentate distinct objects in a scene, and find it's correspondent bounding boxes. Most segmentation methods rely on previous binarization of the image [1] and this one is not different.

In summary, if a pixel has one of it's neighbours (4-connected or 8-connected, for a 2D image), as a foreground

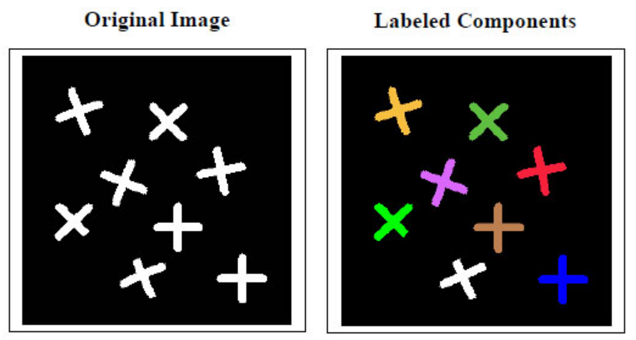

Figure 5. Example of a CCA applied to labeling components in a binary image

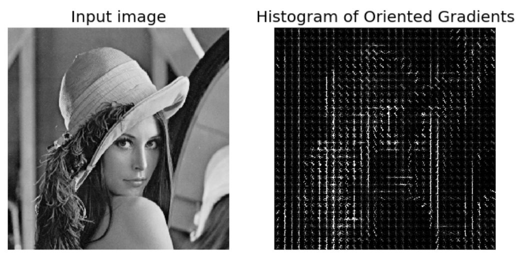

Figure 6. Example of HoG applied to extract features from an image

pixel, this pixel will also be a foreground pixel. It then enumerates all the different objects with distinct labels. The corresponding bounding box is then delimited using the corner pixels of the object.

In this work, we use an 8-connected CCA labeling on multiple binary images.

3.4. Histogram Oriented Gradients

Histogram Oriented Gradients (HoG) is an algorithm used to extract features from an image, which is invariant to geometric and photometric transformations. It is very flexible, as you can change the geometry of the extracting cell and it's size.

3.5. Artificial Neural Networks

Artificial Neural Networks (ANN) is an analogy made from human neurons and how they are connected between each other. It is known that the human brain is actually made of lots of neurons connected in a complex way, and they forward propagate the electric impulses received from multiple "input sensors", and each area of the brain processes these impulses in different ways.

In ANN, to simulate the different areas from the brain, it is assigned a weight to each neuron. We train Artificial Neural Networks to find out which weights are best suited for each classification problem.

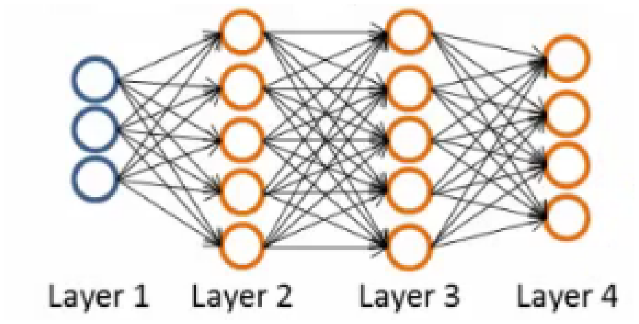

Figure 7. Example of a Artificial Neural Network, with $L = 4$, $K = 4$, $s_1 = 3$, $s_2 = 5$, etc…

To train the ANN, it is first needed to initialize the weights for each neuron. It is very important that these weights are not symmetrical, as this prevents the algorithm to fail. Thus, instead of initializing all weights as zeros, it is recommended to randomly initialize all weights.

Secondly, it calculates the output hypothesis, passing through the input values $(x^{(i)})$, and applying them to the activation functions of each layer l, $(a_l(x^{(i)}))$ multiplying by each neuron weights. This process is called *forward propagation*.

Then, the algorithm evaluates the error caused by each neuron, by evaluating a cost function that compares the output hypothesis and the ground truth. Lastly, it subtracts the error from each weight. This process is called *back propagation*.

This algorithm is computed several times, with a goal to minimize the cost function, given as

$$J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\Big[-y_k^{(i)}\log\left((h_\Theta(x^{(i)}))_k\right) - (1-y_k^{(i)})\log\left(1-(h_\Theta(x^{(i)}))_k\right)\Big] + \frac{\lambda}{2m}\left[\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2\right] \tag{5}$$

where $\Theta_{ji}^{(l)}$ is the weight matrix of layer l, containing the weights for each connection between neuron j in layer l and neuron i in layer $(l+1)$; m is the number of examples; K is the number of labels, or outputs; L is the number of layers; s_l is the number of neurons (or units) in the layer l; and λ is the regularization coefficient.

The rightmost term of formula (7) is called the regularization term, parameterized by λ, and used to avoid overfitting on training data (that is, labels most training data well, but fails on predictions over the test sets).

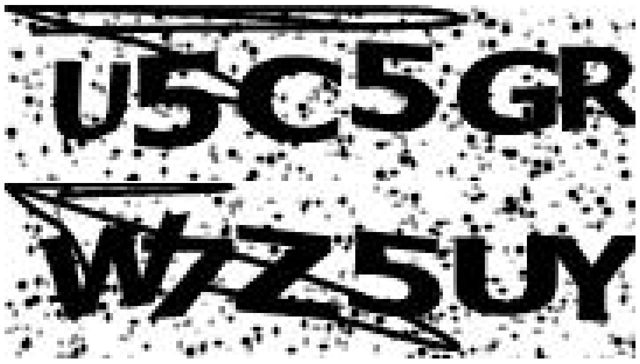

Figure 8. Examples of CAPTCHAS from the benchmark

4. The Benchmark

We created a database from a public website's CAPTCHAs. It consists on RGB, 180x50 pixels, with 6 characters, which some of them are collapsed, some background pepper noise and randomly curved lines crossing characters with the same pixel intensity. Although the images are RGB, visually they are actually grayscale, which, in fact, hampers the binarization. The characters do not vary much of position and height, but they do in width.

The benchmark is filled with 2000 CAPTCHAs of this style, and are manually labeled by humans. When there is uncertainty about any character in the image, this character is labeled as a '?'.

5. System Architecture

First, we train a Character Recognition (CR) ANN with an architecture as follows: 776 features extracted from a HoG algorithm with 7x7 cell window as input [1]; one hidden layer of 150 units; $\lambda = 10$; a maximum of 1000 iterations; sigmoid activation functions for all layers; and 36 output units, corresponding to each digit and letter.

The training set is made of 9793 images of this database characters, with 32x32 pixel resolution, obtained from the segmentation by means of William R. Schwartz algorithm. We obtained a 92% accuracy on the training set and also on the validation set, made of 2116 of different images with the same characteristics from the training set.

Then, the segmentation problem is branched:

5.1. Multi-binarization segmentation

We test several combinations of binarization algorithms and morphological operations over the binarized images, as suggested by [9]. All binarized images passed through a inverting operation, so that characters and noise would be white pixels and background in black pixels.

For each generated image, we apply CCA followed by Non-Maximum Suppression (*NMS*). For this purpose, we use the CR module described above to output the chance of a given image delimited by the bounding box to be a character. We take the maximum probability of all the 36 labels for this. If there is overlap of over a half of any bounding box area, we choose the one that has the biggest probability of being a character. We also utilize the compactness [9] value of the image, as a tiebreaker feature.

Method	*CAPTCHA Accuracy*	*Character Accuracy*	*Proc. Time*
Niblack($N = 41$,$k = 0.4$) Eroded-twice Opened-twice	1.30%	26.10%	893s
Otsu Eroded-once Opened-twice	1.85%	32.57%	1035s
Sauvola($N = 51$,$k = 0.6$) Eroded-twice Opened-twice	1.70%	30.53%	1215s
Otsu Sauvola($N = 51$,$k = 0.6$) Eroded-twice Opened-twice	2.45%	33.78%	2070s

Niblack($N = 11$,$k = 0.0$)
Niblack($N = 11$,$k = 0.2$)
Niblack($N = 31$,$k = 0.6$)
Niblack($N = 41$,$k = 0.4$)
Eroded-twice
Opened-twice
	3.15%	**38.60%**	3916s

Table 1. Results obtained from the multi-binarization segmentation method

If the probability of the bounding box being a character is less than 30%, we also remove it from the candidates, even if there is no overlapping. We make a height boundary too, so that, if the candidate blob has less than 15 or over 35 pixels pixels of height, it is discarded. If the bounding box has more than $1/6$ of the image width, it is also discarded.

To classify the bounding box and find the probability of it being a character, we first have to rescale the segmented image to 32x32 pixels, and then apply HoG on it.

5.2. Hand-crafted method

We noticed that the CAPTCHAs in our benchmark follows a pattern: in every image, the character distribution along the 'x'–axis is the same, and they all have six characters. Thus, we applied a simple segmentation algorithm and obtained relatively good results.

The algorithm simply divides the image in six pieces, each of which has the same sizes for all images. Empirically, a vector is created containing the initial and final bounding boxes positions on the 'x'–axis. And all the the bounding boxes have 34 pixels in the 'y'–axis.

[1]Available at `http://www.vlfeat.org/`

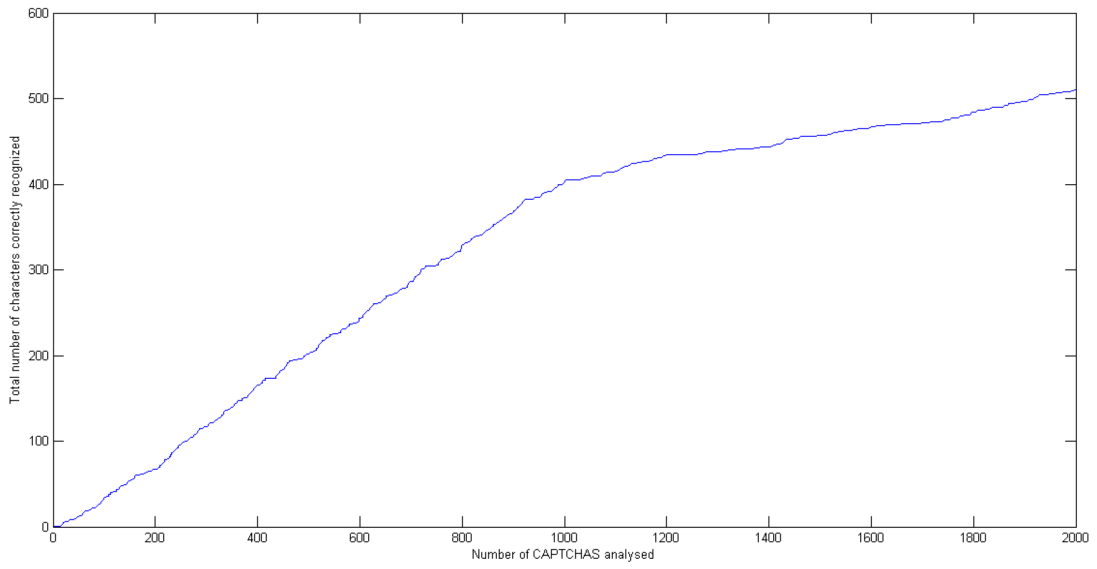

Figure 10. Cumulative character hit count for the multi-binazarition method

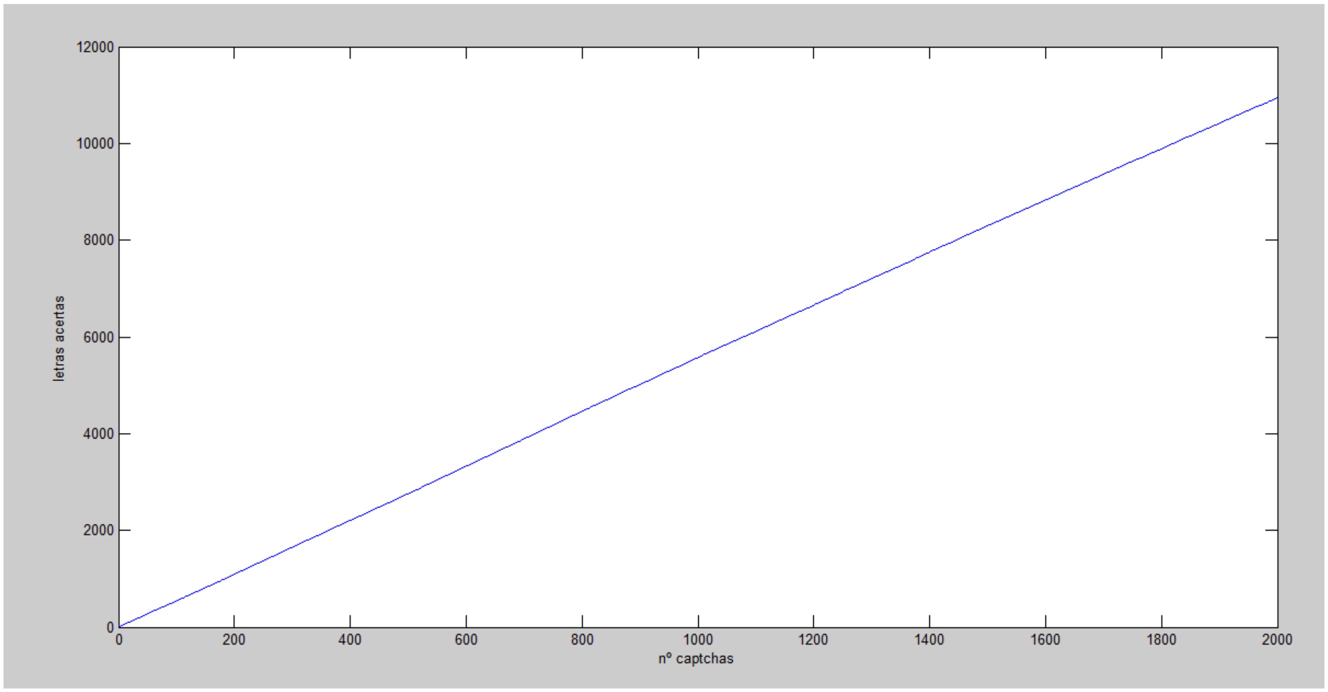

Figure 11. Cumulative character hit count for the hand-crafted method

6. Experiment Results and Analysis

The tests were realized over 2000 CAPTCHAs of the benchmark, each of which containing

Table 8, attached with this document, shows the confusion matrix for the hand-crafted segmentation based method. We note that most of the errors were due to the confusion between *'O' and '0'*, *'V' and 'W'*, *'T' and 'I'* and *'O' and 'C'*. This can be due to the human confusion when labeling the captchas, as there is not a big difference between the caracters in this CAPTCHA database. For the *'O'*

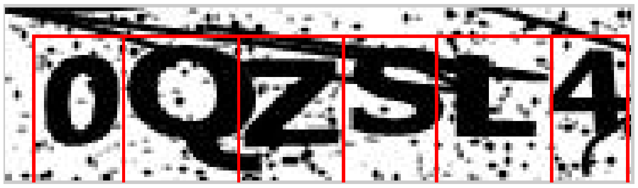
Figure 9. Example of a CAPTCHA segmented with the hand-crafted method

and 'C' case, it can be due to the fact that the last character is usually cropped, or because of overlapping characters.

Type	Accuracy	Hit Count
CAPTCHA Accuracy	59.35%	1187
Character Accuracy	90.44%	10853
Character Accuracy accepting '?'	91.20%	10945

Table 2. Hand-crafted segmentation algorithm accuracies

As can be seen in Figure 10, the CR module starts to decay its efficiency by the 1000 CAPTCHA mark. This is due to the fact that the ANN is trained with character images extracted from the first thousand. Therefore, it is expected that the recognition on the second half of the database would decrease slightly, and perceived.

The best results were obtained from the combination of Niblack($N = 11$, $k = 0$), Niblack($N = 11$, $k = 0.2$), Niblack($N = 31$, $k = 0.6$) and Niblack($N = 41$, $k = 0.4$) binarized images, just as in [9], with it's respective opened-twice images and eroded-once images, obtaining the overall accuracy of 3.15% for CAPTCHA recognition and 38.60% for Character Recognition.

The low Character Recognition hit rate, in relation to the validation set, is due to the training being made with images with different characteristics than those obtained from the proposed method in this article. As they were resized, and may be cropped or augmented in relation to the train database, it may cause confusion to the classifer. To have a higher CR hit rate, the first step would be to make a new character database extracted with this segmentation algorithm, so that it would be possible to make more accurate recognition and thus, improving the overall performance of the system. Our lack of time made this new database creation and labeling impossible.

Now, analyzing the hand-crafted method, we perceive a higher accuracy on both character recognition and, therefore, the end-to-end system accuracy. Thus, proving that the segmentation requires primary attention when developing anti-CAPTCHAs softwares. Therefore, to generate CAPTCHAs that actually fulfill their purposes, they should have a powerful anti-segmentation resistance.

Table 3. Final results comparison

Method	Multi-binarization	Hand-crafted
CAPTCHA Accuracy	3.15%	59.35%
Character Accuracy	38.60%	90.44%
Character Hit Count	1187	10853

7. Conclusions

In this article, we showed what are the best combinations of morphological operations and binarization methods applied to one CAPTCHA database and some cares that should be made when working in similar projects.A thorough analysis of related works would be well suited to obtain better performances on this problem, by other approaches.

We also proved that combinations of binarization is a powerful resource, as they can be applied to general purposes, without the need of previous knowledge of the images' nature.

Finally, we showed that Combination of Niblack's method proved to be the best choice for this purpose, just as in [9],

8. Acknowledgments

We would like to thank Dr. David Menotti for the guidance, the incentive, and all the knowledge provided, so that we could accomplish this task.

References

[1] E. Bursztein, M. Martin, and J. C. Mitchell. Text-based captcha strengths and weaknesses. *CCS'11*, pages 125–137, 2011.

[2] A. K. Jain and S. Bhattacharjee. Text segmentation using gabor filters for automatic document processing. *Machine Vision and Applications*, 5(3):169–184, 1992.

[3] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. *NIPS*, 2011.

[4] W. Niblack. *An Introduction to Image Processing*. NJ: Prentice-Hall, Englewood Cliffs, 1986.

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

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

[7] C. Wolf, J. Jolion, and F. Chassaing. Text localization, enhancement and binarization in multimedia documents. *Proc. ICPR*, 2:1037–1040, 2002.

[8] J. Yan and A. S. E. Ahmad. A low-cost attack on a microsoft captcha. *CCS'08*, pages 543–554, 2008.

[9] Y. Yoon, K. Ban, H. Yoon, J. Lee, and J. Kim. Best combination of binarization methods for license plate character segmentation. *ETRI Journal*, 35(3):491–500, 2013.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	?
0	231	0	0	0	0	0	4	1	2	3	0	1	0	3	0	0	0	0	0	0	0	0	0	0	40	1	3	0	0	0	1	0	1	1	0	0	0
1	0	291	1	0	7	0	0	0	0	0	1	0	0	0	0	0	0	0	9	6	0	5	0	0	0	0	3	0	0	1	0	0	0	0	1	0	0
2	1	0	310	6	0	0	0	1	0	2	0	0	0	2	2	0	0	0	0	1	0	1	0	0	0	3	0	0	0	0	0	0	1	0	0	11	0
3	0	0	1	272	0	0	0	0	3	2	0	10	0	2	0	0	0	0	0	5	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
4	0	4	0	0	324	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	1	0	313	0	0	1	3	0	3	0	1	0	0	0	0	0	1	0	0	0	2	0	0	0	0	7	0	0	0	0	0	0	1	0
6	2	0	0	0	0	2	331	0	1	2	1	3	1	1	0	1	7	2	0	0	0	0	0	1	1	0	2	0	3	0	0	0	0	0	0	0	0
7	0	0	3	2	1	0	0	307	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	11	0
8	0	0	0	8	0	0	1	0	319	0	2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
9	2	0	1	1	1	3	0	0	2	307	0	1	1	2	0	0	0	0	0	0	0	0	0	0	2	2	0	0	5	0	0	0	1	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	351	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
B	0	0	0	4	0	0	1	0	2	1	0	289	0	6	4	1	0	1	1	0	0	0	0	0	0	3	0	4	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	1	0	0	0	260	0	1	0	12	0	0	0	0	1	0	0	9	0	5	0	0	1	0	1	0	0	0	0	0
D	0	0	1	0	0	1	0	0	0	4	0	7	0	304	3	1	0	0	6	0	0	1	0	0	0	4	1	1	2	1	2	0	0	0	0	0	0
E	1	0	1	0	0	2	0	0	0	0	0	4	0	0	304	9	0	0	1	0	0	3	0	0	0	1	0	0	0	1	0	0	0	0	1	1	0
F	0	0	0	0	0	0	0	0	0	0	0	1	0	1	23	279	0	0	2	1	0	0	0	0	0	4	0	2	0	3	0	0	0	0	0	0	0
G	0	0	0	0	0	0	3	0	0	0	0	0	7	0	0	0	316	0	0	0	0	0	1	0	1	0	1	0	2	0	0	0	0	0	0	1	0
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	0	341	4	0	1	0	0	2	0	0	0	0	0	1	1	0	0	0	1	0	0
I	1	12	1	0	1	0	0	0	0	0	0	0	0	0	1	5	0	6	262	2	0	8	0	0	0	0	0	1	0	10	1	0	0	0	0	0	0
J	0	7	0	3	1	0	0	0	0	2	0	0	0	0	0	1	0	0	6	284	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	332	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0
L	0	5	1	0	0	0	1	0	0	0	0	0	0	2	14	0	0	0	4	0	0	283	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	0	0	0	307	9	0	0	0	0	0	0	0	1	1	0	0	0	0
N	0	0	0	0	0	1	0	0	0	0	2	0	0	0	0	0	0	0	3	0	2	0	1	359	0	0	0	0	0	0	0	0	1	0	0	0	0
O	88	0	1	0	0	0	5	0	0	3	1	0	40	6	1	0	11	0	1	0	0	1	0	1	218	0	33	0	2	0	2	0	2	0	0	1	0
P	0	0	1	0	0	0	0	1	1	1	0	7	0	5	2	7	0	3	3	0	0	0	2	0	0	295	0	5	0	0	0	0	0	0	0	0	0
Q	3	1	0	0	0	0	0	0	0	0	0	0	15	1	0	0	2	0	2	0	0	1	0	0	7	0	297	0	1	0	1	0	0	0	0	1	0
R	0	0	0	0	0	1	0	0	1	0	0	2	0	0	1	1	0	3	0	0	3	0	0	1	0	4	2	319	0	1	0	0	0	0	0	0	0
S	0	0	0	2	1	7	1	0	4	3	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	301	0	0	0	0	0	0	1	0
T	1	7	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	20	1	0	0	0	0	0	0	0	0	0	311	0	0	0	0	1	0	0
U	3	0	0	0	0	0	0	0	0	2	0	0	0	2	0	0	0	1	0	0	0	3	1	0	1	0	0	0	0	0	340	0	0	0	0	0	0
V	1	1	0	0	0	0	0	3	0	0	0	1	1	0	0	0	0	0	0	2	0	0	2	1	0	0	0	0	1	1	1	287	19	0	5	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	21	249	0	0	0	0
X	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	1	0	0	0	1	0	327	2	0	0
Y	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	5	0	0	0	0	0	0	2	0	4	0	0	356	0	0
Z	0	0	8	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	277	0
?	0	1	1	2	3	1	3	2	0	0	0	7	10	2	0	1	2	1	3	3	0	0	0	0	3	2	3	0	1	5	3	16	15	0	0	2	0

Table 4. Confusion Matrix generated using the hand-crafted segmentation method