

LICENSE PLATE NUMBER RECOGNITION

New Heuristics and a Comparative Study of Classifiers

César García-Osorio, José-Francisco Díez-Pastor, Juan J. Rodríguez and Jesús Maudes
Higher Polytechnic School, University of Burgos, Avda. Cantabria, Burgos, Spain

Keywords: Computer Vision, Digital Image Processing, Optical Character Recognition, Smoothing Spatial Filter.

Abstract: We describe an artificial vision system used to recognize the Spanish car license plate numbers in raster images. The algorithm is designed to be independent of the distance from the car to the camera, the size of the plate number, the inclination and the light conditions. In the preprocessing steps, the algorithm takes a raster image as input and gives an ordered list of license plate areas candidates. Only in very rare occasions the second and third candidates are needed; most of the cases the first candidate is the correct one. During the preprocessing a new filter has been used. This filter is only applied to low saturation areas, getting better results this way. In order to choose the best classifier for the classifying stage, a comparative study has been performed using the data mining tool Weka.

1 INTRODUCTION

License plate number recognition or license plate recognition (LPR) is one of the most practical application of image processing and pattern recognition techniques. Its most common use is the control of parking areas (Sirithinaphong and Chamnongthai, 1999) and traffic monitoring (Setchell, 1997). In these contexts the distance to the camera, position, inclination and other variables that affect the image are quite well controlled and hence it is possible to take advantage of these restricted working conditions (Emiris and Koulouriotis, 2001). The application presented in this paper is designed to work in not so restricted and structured environments. We want to locate the license plate and recognize its number independently of its size, orientation, position or lightening condition. The only restriction we assume is that the background of the license plate number is white and the characters color is black. This is the common style for the Spanish license plates which we want to recognize.

The process of license plate number recognition involve three main steps: i) license plate localization, ii) character segmentation, and iii) character recognition. We have structured the rest of the paper in accordance with these steps. However, we will give an introduction to the Spanish licence plate number system first.



1.1 Spanish License Plates Formats

The increase in the number of cars has put to the limits the license plate number systems. So, plate number systems have needed to be adapted as the number of cars approached the maximum number these systems could register. In Spain, initially, the plate number were composed by one or two letters to code the province followed by up to six digits. This system lasted till 1971 when the plate numbers in Madrid (Urios-Mondéjar, 2007), the capital of Spain, were approaching the number 999999. In the new system there were four digits and a letter in a first moment, and later was necessary to use two letters. With the joining of Spain to the European union a new change happened.

In the year 2000, and close again to the coding limit of the system, a new change is made. In the new system a plate number is composed by four digits followed by three letters. Besides, the province code disappeared and an European Union logo with blue background and a 'E' in white¹ appeared in the left side of the plate number. As well, whenever a plate number need to be substituted because it is deteriorated, the new plate number will come with the European Union logo, even if the plate number is in the old format with the province code.

¹ 'E' for "*España*", "Spain" in Spanish

Table 1: Common Spanish plate number formats

With province code and no letter	V· 298176
With province code and letters	V·1257·HJ
With province code, letters and European Union logo	 BU 7893 T
Current plate number code	 0802 FXX

In Table 1 it is possible to see all these formats². These and others facts about Spanish plate numbers can be found in the webpage <http://www.geocities.com/amoros29/>.

2 LICENSE PLATE LOCALIZATION

2.1 Image Preprocessing

The first stage of the process is to locate the licence plate area in the input photo. In order to do so, we need to emphasize regions of high spatial frequency that correspond to edges. Therefore, we first use the classic Sobel filter. As the input to this step of the process is an RGB color image, and we know that in the license plate the background is white and the characters are black, we will apply the horizontal Sobel filter to the areas of low saturation. The saturation is obtained from RGB color image using the formula

$$S = \begin{cases} 0 & \text{if } \max\{R,G,B\}=0, \\ \frac{\max\{R,G,B\}-\min\{R,G,B\}}{\max\{R,G,B\}} & \text{otherwise.} \end{cases} \quad (1)$$

All the achromatic colors (grey scale colors) have the same saturation and lower than any chromatic color. Note that the white could be actually gray depending on illumination conditions. The black and white pixels of the license plate areas will have low saturation. So, we discard all the pixels with high saturation and apply the Sobel filter only to the pixels with low saturation. Experimentally, we found that the best results are obtained when applying the Sobel filter to pixels with saturation lower than 0.35. In Figure 1 we can see the results of using this heuristic.

²In the Spanish plate number system there are two rows plate numbers too, as well as other special plate numbers, that has not been taken into account in the applications explained in this paper.



Figure 1: Top left: Original RGB image. Bottom left: After applying traditional Sobel filter. Top right: the image without the low saturation areas. Bottom right: After applying the conditional Sobel filter.

Another heuristic we use here is to discard the 30% top part of the image and the 10% of the bottom part. This areas never have a license plate number (Rodríguez, 1999).

Next in the process is the binarization. We obtain a black and white image from the result of the previous transformation. The binarization threshold is chosen to get a black and white image with only 2% of the pixels being white, this way only those pixels with high intensity variation are considered as edges in the license plate. The edges due to background, or without that high intensity variation are discarded.

After all the previous preprocessing, the license plate will appear as a sequence of white and black transitions with a lot of white pixels concentrated in the license plate area. In other parts of the image the concentration of white pixels is lower. As the last step of this stage we use an horizontal soften filter. This filter can be thought as a kind of mean or average filter. The idea is to count the number of white pixel in a rectangular area with center at each pixel. Empirically, we found the best results were obtained with a size 3x61. If the number of white pixels is greater than 50 we leave that pixel white. That is, $M(x,y) = 1$ (1=white), if $\sum_{i=-30}^{30} \sum_{j=-1}^1 M(x+i,y+j) > 50$.

When we apply this filter to a line with a lot of low concentrated edge pixels, all those pixels are eliminated. On the contrary, when we apply this filter to a line with less edge pixels but high concentrated, the line will became white.

One could think that the effect of applying this filter is the same as the dilation/erosion of mathematical morphology (Serra, 1982), but this filter seems to be less sensitive to noise. In Figure 3 we can see the results of dilation/erosion with different window sizes.

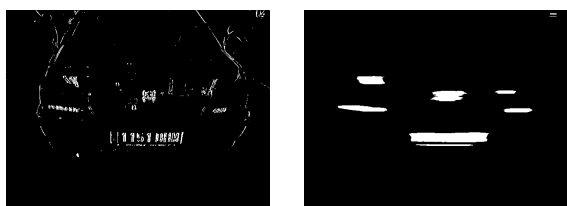


Figure 2: Left: Image after binarization. Right: Image after applying the horizontal soften filter.



Figure 3: Left: Image after dilation/erosion with a window 15 pixels width. Right: Image after dilation/erosion with a window 20 pixels width.

2.2 Selection of Plate Candidate

After applying the last step of the previous stage, we have a black image with one or more white areas candidate to be a license plate. However, only one of those areas corresponds to the real license plate, while the others are consequence of noise or other objects in the background. Now, we need an algorithm to determine which one of these white areas is the real plate.

We have used several heuristics. First, we only keep the ten biggest areas, this way we eliminate most of the areas due to noise. We then order those ten plate candidates according to the following criteria

- ratio of width to heights greater than two, since a license plate is wider than taller.
- distance to the center of the image, the areas far from the center are unlikely to be plates.

For two areas with similar characteristics we choose the one with lower positions, we do this because we have found that some times the area due to the edges in the car radiator will give end as a false positive plate.

Note, that we keep all the ten areas not discarded. Later, if in the following stages we realize that the selection of plate area was incorrect we recover the following plate candidate from the ordered list. In the experiment this only happened in few occasions.

3 CHARACTER SEGMENTATION

After the previous stage, we obtained a license plate image. We need to do further image processing before



Figure 4: Top left: Original RGB image. Bottom left: After applying traditional Sobel filter. Top right: After binarization. Bottom right: After rotation.

starting to locate the characters in the plate. We have not tried any new heuristic with these steps. We apply the classic Sobel filter (Gonzalez and Woods, 2002), gradient binarization (Rodríguez, 1999) and Hough transform (Gonzalez and Woods, 2002) to correct the plate inclination if needed (see Figure 4).

3.1 Character Localization

Now with the preprocessed plate we use a novel method to locate the characters in the plate. That is one of the most delicate steps of the process.

First, we try to adjust further the license plate and we eliminate from the border of the image all the rows and columns with only white or only black pixels.

Second, we apply a coarse method to discard those blobs of black pixels which are obviously consequence of noise and are not really characters. This process must be conservative, because we do not want to discard correct characters. We have follow several heuristics:

- We discard all the areas whose width is greater than 1/8 of the width of the area we have identified as a license plate candidate. As the license plate will have a minimum of seven characters, an area with bigger width will be for sure not a character.
- We discard as well the areas which are taller than 0.7 the height of the plate.
- We use the color information to eliminate the areas with at least 50% of high saturation pixels. This areas correspond to the blue European Union logo. This have the effect of eliminate other chromatic areas in case our location of the license plate would have not adjust close enough to the plate.
- If the width of the widest area is lower than 1/20 of the width of the area we have identify as the license plate, we have found a case of false positive. We need to start the process with the next candidate from the list of plate candidates.

Next, we apply a more refined strategy. A character will be a blob of connected black pixels (or several, if the characters parts have been disconnected as a consequence of the previous preprocessing) fully inside of a rectangular area of dimensions calculate as a



Figure 5: License plate before and after eliminating the black areas that does not correspond with characters.

percentage of the dimensions of the license plate area. We align in turn the top left corner of this rectangle with the top left corner of all the blobs, starting with the one in the very top left. Be β_1 the blob we have used to align the rectangle. It will not be the case that β_1 does not fully fix inside the rectangle, because all that big blobs have been discarded with the previous coarse method. Now, if there is piece of another blob β_2 that is partially inside the rectangle, but not fully inside, we can discard β_1 . It will not be a character or part of a character, β_1 is probably noise. If there are not others blobs partially inside the rectangle, and there is another blob β_2 fully inside the rectangle, we can consider both β_1 and β_2 as parts of the same character. This method is able to group pieces of characters that have been disconnected after applying the filters, and at the same time discards any remaining noise.

In Figure 6 we can see an example of the method. The coarse method was not able to eliminate all the noise. The plate border is broken and was not big enough as to be discarded. Besides the “U” appears broken in two pieces. In the proposed method the rectangle will be aligned with the top left pixel of the first piece, and as the other piece is fully inside the rectangle both pieces are identified as part of the same character. On the contrary, the small blob due to the number plate screw is not consider part of the character because when we align the rectangle with its top left pixel, the six is only partially inside the rectangle. The same happens in the right side, when we align the rectangle with a border, and the “Z” is only partially inside the rectangle.

We have found this method slightly better than the traditional horizontal projection method for character segmentation and less sensible to broken characters. Still, in same rare circumstances we could group some scattered points and identify them as parts of characters. But this cases could be easily managed using as criteria the number of black pixels.

3.2 Character Normalization

Before using a classifier to identify the characters we need to normalize them. There are three reasons for normalization:

- We can reduce the number or attributes and that will make the learning of the classifier faster.

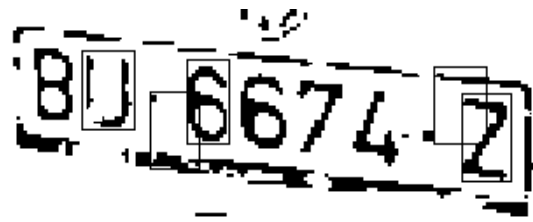


Figure 6: Character segmentation.

- We will have the same number of attributes for all the characters because they will have the same dimensions.
- We will improve the efficiency of the classifier as the input is more similar to the others instances in its class, and more different from the instances in other classes.

The normalization steps are:

- Center the character.
- Resize it to 16×16 keeping its ratio width/height (this is specially important for characters like “I”).

4 CHARACTER RECOGNITION

In this last section, we analyze and compare which set of characteristics and classifiers give the best classification results. To do this study we use Weka (Witten and Frank, 2002), a data mining tool written in Java <http://www.cs.waikato.ac.nz/ml/weka/>. We evaluate the performance of seven different classifiers and four different sets of characteristics using 2070 characters from about 700 licence plates.

We use 10×10 -fold cross validation, in each of the experiments 90% of the instances were used for training and 10% for validation, and this was repeated another nine times, with different 10% of instances each.

An important characteristic we could extract from the character image is the number of holes in the characters. That increase the classification accuracy for characters like “P” and “F” or “B”, “8”, “6” and “9”. The sets of characteristics use in this study were:

- In this data set we represent each of the 2070 characters using the whole $16 \times 16 = 256$ pixels of the character image together with the number of holes: 257 attributes in total. This is basically the character without any characteristic extraction process.
- We use the horizontal and vertical projections of the character. This gives 16 characteristics from the horizontal projections, 16 characteristics from

the vertical projections, plus the number of holes: 33 characteristics.

- c. In this data set each character is represented by its main four Kirsch compass filters, horizontal, vertical, and both diagonals. The results are down sampled to a 4×4 image, and the original image is down samples as well. This gives $5 \times 16 = 80$ characteristics, plus the additional one for the number of holes: 81 characteristics.
- d. We use a overlapped down sample of the original image. Each 4×4 block of pixels gives a value, then this mask is moved two pixels right, and so till the right side, then the mask is lower two pixels and the process repeated from the left. We obtain 49 attributes, plus the number of holes: 50 characteristics.

Regarding the classifiers, we tried the following:

1. functions.MultilayerPerceptron, with parameters: GUI=false, autoBuild=true, Debug=false, Decay=false, HidenLayers=(number of attributes + number of classes)/2, LearningRate=0.3, momentum=0.2, nominalToBinaryFilter=true, normalizeAtributes=true, normalizeNumericClass=true, RandomSeed=0, trainingTime=500, validationSetSize=0 and validationThreshold=20. Neural Networks has been used as classifiers in (Draghici, 1997; Nijhuis et al., 1995)
2. bayes.NaiveBayesUpdateable, with options: debug=false, useKernelStimator=false and useSupervisedDiscretization=false.
3. functions.SMO, Platt's sequential minimal optimization algorithms for training Support Vector Machines (Platt, 1999; Zheng and He, 2006; Chengwen et al., 2006), with options: buildLogisticModels=false, C=1.0, checksTurnedOff=false, Epsilon=1.0E-12, filterType=Normalize training data, Kernel=PolyKernel, RamdomSeed=1 and toleranceParameter=0.0010.
4. lazy.IBk, with no distance weigthing and options: KNN=1, crossValidate=false, Debug=false, MeanSquared=false, NearestNeighbourSearchAlgorithm=LinearNN, WindowsSize=0.
5. meta.AdaBoostM1, with options: classifier=J48, Debug=false, NumIterations=10, useResampling=false, WeightThreshold=100.
6. meta.Bagging, with options: classifier=J48, Debug=false, NumIterations=10, bagSizePercent=100, calcOutOfBag=false, Seed=1.
7. trees.J48, with options: binarySplits=false, ConfidenceFactor=0.25, debug=false, MinNumObj=2, numFolds=3, reducedErrorPruning=false,

Table 2: Results comparison

	a.	b.	c.	d.
1	99.10	97.88	98.15	99.03
2	<u>96.42</u>	<u>89.44</u>	<u>89.86</u>	<u>93.65</u>
3	99.19	<u>96.85</u>	98.01	99.09
4	<u>97.86</u>	<u>96.38</u>	<u>96.45</u>	<u>97.74</u>
5	<u>97.56</u>	<u>95.39</u>	<u>96.23</u>	<u>96.81</u>
6	<u>95.46</u>	<u>92.92</u>	<u>93.87</u>	<u>95.94</u>
7	<u>94.04</u>	<u>89.17</u>	<u>91.04</u>	<u>93.66</u>

SavaInstanceData=false, Seed=1, subtreeRaising=true, Umpruned=false, UseLaplaze=false.

The results of the experiments are shown in Table 2. The underline values indicate a significant worse accuracy compares with the multi layer perceptron that has been used as the base of the comparison. We can see that all the classifiers have an accuracy higher than 90% for all the data sets, what shows the benefits of all the preprocess and normalization work. Note, thought, that the best results are obtained when the character is used without any characteristic extraction (a). The worse set of characteristics are the horizontal and vertical projections (b).

Regarding the classifiers, as we could expect, AdaBoost and Bagging improve the performance of the J48 that they are using as base learner, being AdaBoost better than Bagging. The best classifiers is the SMO using directly the binary matrix that represent the character. However, finally we decide to use the multi layer perceptron, it is only a bit slower but is faster and its memory requirements are more than three times lower.

5 CONCLUSIONS

We have described an artificial vision system used to recognize the Spanish cars license plate numbers in raster images. We combine the use classic image processing techniques with some new ideas, such as the soften filter and the character segmentation method.

In the study of classifiers we confirmed the benefits of the preprocessing stages achieving accuracies above 90% for different sets of characteristics. For two of the classifiers we increase the accuracy to 99%.

In future version, we expect to take into account the two row Spanish license plates, and the special license plates with different combination of background and foreground colors.

As well, as one of the reviewers suggested, we want to prepare a more detailed study of the use of

the soften filter, specially regarding its robustness to noise in comparison with other filter in the literature.

ACKNOWLEDGEMENTS

This work has been possible thanks to the project BU004B06 from the “Consejería de Educación de la Junta de Castilla y León”, Spain.

REFERENCES

- Chengwen, H., Yannan, Z., Jiaxin, W., and Zehong, Y. (2006). An improved method for the character recognition based on svm. In *24th IASTED International Conference on Artificial Intelligence and Applications*, pages 457–461, Anaheim, CA, USA. ACTA Press.
- Draghici, S. (1997). A neural network based artificial vision system for license plate recognition. In *International Journal of Neural Systems*, volume 8, pages 113–126.
- Emiris, D. E. and Koulouriotis, D. E. (2001). Automated optic recognition of alphanumeric content in car license plates in a semi-structured environment. In *International Conference on Image Processing, ICIP*, volume 3, pages 50–53.
- Gonzalez and Woods (2002). *Digital Image Processing*. Addison-Wesley, Reading, MA, USA.
- Nijhuis, J., Brugge, M., Helmholt, K., Pluim, J., Spaanenburg, L., Venema, R., and Westenberg, M. (1995). Car license plate recognition with neural networks and fuzzy logic. In *IEEE International Conference on Neural Networks*, volume 5, pages 2232–2236. ACTA Press.
- Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In Scholkopf, B., Burges, C., and Smola, A., editors, *Advances in kernel methods: support vector learning*, pages 185–208. MIT Press.
- Rodríguez, F. M. (1999). *Contribución al reconocimiento de caracteres en imágenes complejas*. PhD thesis, Escuela Técnica Superior de Ingenieros de Telecomunicaciones, Campus Universitario, 36310, Vigo, Galicia, Spain.
- Serra, J. (1982). *Image analysis and mathematical morphology*. Academic Press, London, 2nd edition.
- Setchell, C. J. (1997). *Applications of Computer Vision to Road-traffic Monitoring*. PhD thesis, Faculty of Engineering, Department of Electrical and Electronic Engineering.
- Sirithinaphong, T. and Chamnongthai, K. (1999). The recognition of car license plate for automatic parking system. In *Fifth International Symposium on Signal Processing and Its Applications, ISSPA*, volume 1, pages 455–457.
- Urios-Mondéjar, D. (2007). Sitio de las matrículas españolas. <http://www.geocities.com/amoros29/>. (last visited: 2-Dec-2007).
- Witten, I. H. and Frank, E. (2002). *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufman, 500 Sansome Street, Suite 400, San Francisco, CA 94111, 2nd edition.
- Zheng, L. and He, X. (2006). Number plate recognition based on support vector machines. In *IEEE International Conference on Video and Signal Based Surveillance (AVSS'06)*, page 13, Washington, DC, USA. IEEE Computer Society.