AUTOMATIC CAR PLATE RECOGNITION USING A PARTIAL SEGMENTATION ALGORITHM

Fernando Martín, David Borges.
Signal Theory and Communications Department. Vigo University.
E.T.S.I.T. Ciudad Universitaria S/N. 36200 Vigo (Pontevedra). Spain.
Phone: +34-986-812151. Fax: 986-812116. E-mail: fmartin@tsc.uvigo.es.

ABSTRACT

In this paper we present a new method for car plate recognition. In the past [1] [2], our group has worked extensively in this problem and one of our main conclusions was that segmentation was responsible for most part of the system errors. This upgrade is based on the simple idea of not performing a full segmentation in order to reduce errors. As we will explain along the paper, the new method is very simple and its results are quite good (better than those achieved with full segmentation).

KEY WORDS

Machine Vision, Mathematical Morphology, Optical Character Recognition.

1. INTRODUCTION

The purpose of our research is the automatic recognition of car license numbers. The main prerequisite is obtaining good car images. The paper in [1] describes how to deal with this problem with moving cars and variable environment conditions. In this paper we will focus on the processing problems that arise after the plate has been located and binarized. Plate location is based on mathematical morphology [3], the particular algorithm has already been published [4] and we only provide a brief outline here. For binarization we use the standard method from Otsu [5].

In the literature, there are many publications about systems of this kind [1] [2] [6] [7] [8]. All of them get their results through a process of location, segmentation and recognition. The key detail here is to remove part of the segmentation process doing only a "partial" segmentation and thus avoiding a great part of the segmentation errors. As we will see in section 3, this simple idea will yield very good results. The price to be paid will be a slower recognition process (although we have tried to optimize it as much as possible).

2. ALGORITHM DESCRIPTION

2.1. Plate Location

This method for plate location uses the following model: "characters are black objects of small thickness within a white background". A morphological transformation called "top-hat" (more exactly a "black top-hat" or "bottom-hat") will help us to find characters. We will use a circular structuring element (S.E.) with diameter bigger than character stroke width. This will make sure that the bottom-hat enhances the characters (the operation is $\text{closing}(I) - I$, so that it will work if the closing is able t erase the characters).

Figure 1. Car image (left) and "Bottom hat" (right).

To completely locate the plate, we will binarize the bottom-hat result so that we can work with binary morphology (using the classical Otsu method [5]). I we continue making a closing with a suitable linear S.E. (a horizontal line wider than character spacing) we will convert the characters into a white rectangle, fig. 2.

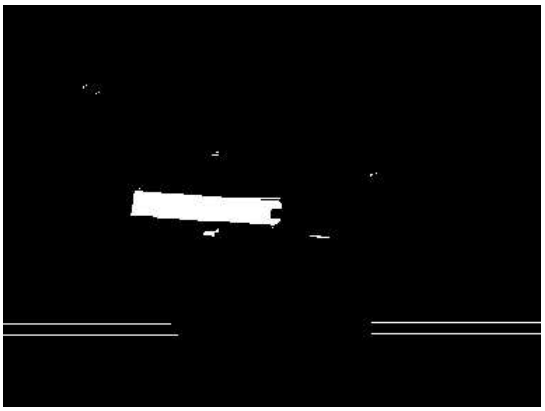

Figure 2. Characters are joined into a rectangle.

We have to apply other morphological operations to filter out all the other white objects in the image (noise). We will use size information until we have the elementary rectangle, then we make it a big bigger so as not to loose any point (fig. 3).

Figure 3. Location result.

An especial case is that of square plates. In these cases, we get two regions. When we finally dilate both rectangles they are joined together (fig. 4).

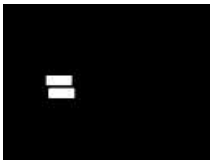

Figure 4. Square plate.

After testing this algorithm, we found that it was able to find 96% of the plates. Sometimes, the algorithm produces also false outputs (the next stage will be able to reject them).

This method depends strongly on character size (all S.E. sizes can be computed from expected character size). If the images are taken at a fixed distance, there is no problem.

The method we have explained is only valid for dark characters on bright backgrounds. To locate bright characters is necessary to perform a white top-hat ($I - \text{opening}(I)$). The system must search for dark characters first and, if it finds nothing, search for bright ones.

2.2. Partial Segmentation

From now on, we will assume that character pixels are black and that background is white. See that we can have plates that are contrary to this, but we know if a plate is so (reversed plates are detected by a white top-hat) and we compute the negative image for reversed plates. We also classify plates on square and rectangular ones depending on aspect ratio (rectangular if ratio is bigger than 3).

2.2.1. Initial Steps:

We will classify plates in three classes according to borders (fig. 5). These are:

a) Borders encircle all characters.
b) A small part of the border is erased.
c) A small part of the border remains.

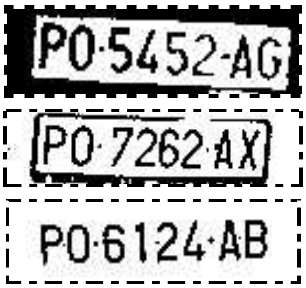

Figure 5. Examples for the tree types (left a, middle b, right c). Dotted border has been added.

The classification method is the following:

1. We label the white objects (connected pixel sets) of input image. The biggest object will be the background. If this object is big enough for enclosing all characters and does not touch the image limits we have a "type a" plate. An object can enclose characters if it is wider than 6 times character minimum width (4 times if plate is square) and if it is taller than character minimum height (twice that value for a square plate).
2. Now, we label the black objects. The widest object must be part of the border. If it is wide enough to enclose all characters we have a "type b" plate. Otherwise, we have a "type c" one.

Border removing is as follows (according to type classification):

a) In this kind of plates is easy to find the border using the horizontal contour of the labeled background object. We have to pay attention to individual characters that could be touching the border.
b) In this case we have labeled the border. We use its vertical contour to erase the vertical portion. All the connected black objects that are too wide to be a character are also erased. In this kind of plates is less likely to have characters touching the borders, because borders are weaker now.
c) In this case, the border is almost erased. We simply erase all the connected black objects that are too wide or too high to be a character.

After border removal, we segment the two rows of the square plates using the vertical projection minimum as the break point.

Remember that we could have more than one possible plate from the location system. We use heuristic conditions to detect false plates:

1. Plate area must be more than 7 times the character minimum area.

2. The number of valid characters must be at least 6 and at most 15 (in the following subsection, we will see how to count the characters).

2.2.2. Computation of Character Ending Marks:

Our partial segmentation idea is based on detecting the x coordinates where each character ends. These character ending marks will define character searching regions (characters will be recognized with a correlation algorithm that recognizes and searches characters at the same time).
Marks are computed using the correlation between each two neighbor columns. These quantity should become zero after each character. We also add a mark at the beginning of the first character (first point where the correlation becomes positive).

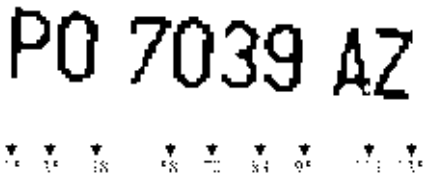

Figure 6. Character ending points.

Some times we have problems with linked characters and/or false characters. We will study more slowly those cases where the distance between two consecutive marks is too big (bigger than the sum of maximum character width and maximum character spacing). We compute the horizontal trace (projection) of the black blobs. If we find a notorious (not null) minimum, we can conclude that we have two linked characters (fig. 7). If we do not find such a minimum, probably we will have a false character (fig. 8).

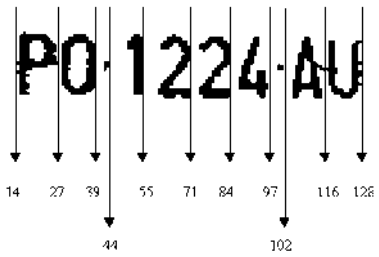

Figure 7. Linked pair of characters.

Figure 8. False character "E" is erased (the character comes from a country label, above image).

We have estimated that 3% of plates get wrong characters at one of the plate two ends (most of them are erroneous **"I"** characters due to imperfect border removing, fig. 8).

2.3. Recognition

Characters are found and recognized between each two marks using correlation with an alphabet of known patterns. Both patterns and input images are binary ("0" for the white background, "1" for the black character points –foreground-) so that correlation is only a sum (much faster). Besides, we perform two key modifications on basic correlation:

1. We divide the result by the pattern area, so that we obtain a normalized correlation. We will consider as valid candidates those patterns with a normalized correlation that is above some predefined threshold. We found the optimum threshold to be 0.4.
2. If some black point in one of the images is not near another black point in the other image, we will subtract a predefined constant (penalty constant) from the total correlation value. This avoids our system to find a **"P"** inside a **"B"** (fig. 9). We have optimized the penalty constant to a value of 2.

In order to get better coincidence, we allow our pattern to get positions "a little beyond" the character ending mark (fig. 10). Remember that we are not breaking the plate.

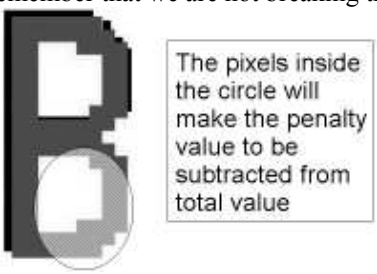

Figure 9. Penalty constant.

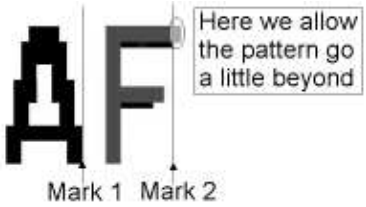

Figure 10. Pattern movement.

2.4. Post-recognition Stage

At this moment, we could have a recognition result, selecting all the candidates with maximum normalized correlation (if bigger than threshold, if not the blobs between those marks are not valid characters).
To improve further the result, we have included some intelligence here. We know that plate characters are grouped into groups that are separated. Each group is formed by letters only or numbers only but not both. We

use the maximum correlation candidates to qualify each group (deciding letters or numbers by majority), then we remove all candidates that do not match the group rule. We sometimes (when the plate layout is appropriate) can also apply the spanish rules for plates, id EST: the province prefixes, this helps to remove the erroneous characters that we commented on section 2.2.
In these conditions our system is able to recognize correctly 97.91% of the characters.

3. RESULTS

3.1. Error Rate & Real Time Tests

The system was tested on a database of 98 car images obtained with a digital camera. The global recognition rate was of 91.83% (less than 8% error rate). Average execution time was of 3.05 seconds per image (the computer was an AMD K6-2, 450 MHz).

3.2. Comparison to Other Systems

Comparing this system to our previous systems of [1] and [2] the segmentation removing has also removed a great part of the errors (these previous work had never been over 85% success rate).
System on [6] uses syntactical pattern recognition (and region growing for plate location). Syntactical recognition seems not very suitable for this problem as is very sensitive to noise. Systems on [7] and [8] do not publish their global results (although partial results in [8] are similar to those in [1] and [2] as is a segmentation dependant system).
We can also compare this system to some commercial ones. The SIAMS 600 system claims to have a low global error rate of 2.5% (being the best system that we know). Another system from Recogniform Technologies spends 5 seconds on recognizing a car, getting a success rate of 90%.

4. CONCLUSION

The main conclusion is that the idea of removing segmentation (although only partially) removes most of the errors. See that recognition is as simple as a correlation (in front of the neural networks of [1] and [2]). The advantages in simplicity and efficiency are clear over previous work in this field.
Practical applications of these systems are almost infinite: traffic control, automatic toll payment, stolen car detection, in/out control in parking lots or buildings...

5. FUTURE LINES

Future working lines for this system are summarized as follows:

- Removing even the "partial" process of segmentation, doing no segmentation at all (knowing that "partial" segmentation is still responsible for an important part of the errors).
- Testing more advanced methods for the recognition process. For example, translated Haussdorf distance [9].

REFERENCES

[1] X. Fernández, F. Martín et al, Automatic and Real Time Recognition of VLP's (Vehicle License Plates), *Lecture Notes on Computer Science, 1311*(2), 1997, 552-559.
[2] F. Martín et al, New Advances in Automatic Reading of VLP's, *Proc. SPC-2000 (IASTED),* Marbella, Spain, 2000, 126-131.
[3] R. C. González, R. E. Woods, *Digital Image Processing* (Reading: Addison-Wesley, 1992).
[4] F. Martín et al, Un Nuevo Método, Basado en Morfología, para la Localización de Matrículas, *Proc. URSI-00*, Zaragoza, Spain, 2000, 139-140.
[5] N. Otsu, A Threshold Selection Method for Gray Level Histograms, *IEEE Transactions on System, Man and Cybernetics,* 1979.
[6] J. R. Cowell, Syntactic Pattern Recognizer for Vehicle Identification Numbers, *Image & Vision Computing*, 1995.
[7] F. X. Jové et al, Sistema Automático para el Reconocimiento de Matrículas, *Proc. URSI-97*, Bilbao, Spain, 1997, 185-188.
[8] J. Barroso et al, Number Plate Reading Using Computer Vision, *Proc. International Symposium on Industrial Electronics (ISIE-97),* Guimaraes, Portugal, 1997.
[9] J. Rucklidge, Efficiently Locating Objects Using the Hausdorff Distance, *International Journal of Computer Vision*, 1997, 251-270.