A System for the Automatic and Real Time Recognition of V.L.P.'s (Vehicle License Plate).

Xulio Fernández Hermida, Fernando Martín Rodríguez, José Luis Fernández Lijó, Fidel Pita Sande, Miguel Pérez Iglesias.
Departamento de Tecnologías de las Comunicaciones. E.T.S.I.T. Universidad de Vigo.
Ciudad Universitaria s/n. 36200 Vigo (Pontevedra). Spain.
Phone: +34-(9)86-812131. Fx: +34-(9)86-812116.
E-mail: xfernand@gpi.tsc.uvigo.es, fmartin@gpi.tsc.uvigo.es, joseluis@gpi.tsc.uvigo.es, fidel@gpi.tsc.uvigo.es, miguel@gpi.tsc.uvigo.es.

Abstract:

In this work, we describe a system able to recognize the V.L.P. (Vehicle License Plate) of a car from an image of it. The system is still under development and has many practical applications. As examples, we can mention: parking accounting systems, traffic monitoring and security systems of many kinds. The system, at an experimental stage, is installed in a parking at the time of writing.

1 Introduction:

The purpose of the project is the recognition of the V.L.P., having as starting point an image of the car. We don't want to make assumptions about the size, and color of the plate (furthermore we use gray-scale images). We distinguish among the following kind of plates:

- Form Classification: there are plates with one row of characters (rectangular plates) and plates with two rows of characters (square ones).
- Background Color Classification: there are plates with dark characters on a bright background (like Spanish ones) and other plates with bright characters with a dark background (like Portuguese ones). In the latter case we compute the negative of the image (we reduce these plates to the former case).

With this assumptions in mind we can explain the stages the total algorithm was divided in:

- Character Location: locating all the zones in the image that may contain characters [1]. Sometimes, we will locate zones with no characters at all; and other zones (like labels and trade marks) that have nothing to do with the plate. The method for character location is based on the derivatives of the image.
- Binarization: choosing a threshold that distinguishes between black and white (in a gray-level image). We proved a widely used algorithm from [2] and also designed an "ad-hoc" algorithm (described in section 2.2).
- Correction of a possible skew angle: computation of that angle and rotating of the image. We basically used the algorithm described in [3].
- Segmentation: dividing the possible zones into individual characters. This is performed by an "ad-hoc" adaptive algorithm. In this stage, we apply a set of heuristic rules to decide which of the possible zones is the true plate.

- The final stage is an O.C.R. We used a template-matching method based in the comparison of the horizontal and vertical projections of the characters [4].

Another important issue is that of getting good images for a proper recognition. We can speak about:

- Considerations needed to choose and configure the camera and to choose the positions and characteristics of the focus (or focuses).
- The minimal resolution needed for the algorithms to work.

We will proceed now to explain in detail each of the stages above. After that, we will comment the practical results.

2 Recognition Algorithms:

2.1 Character Location:

Fig. 1. Image of a car with a recognizable V.L.P.

In this part, our starting point is the image of a car with a recognizable V.L.P. (recognizable by a human observer). The purpose of this stage is to locate the plate that encloses the license number.

To search for the plate we will try to model every character zone present in the image. Our model is based on derivatives. We compute the horizontal Sobel gradient of the whole image. The character zone presents high positive values followed by high negative ones. We look for rectangular zones in which we have a repetition of this gradient pattern. The distance between the positive values and the negative ones must be approximately constant (and this distance is an estimate of the character width).

Our algorithm tries to find the possible rectangles that enclose the plate. We have implemented many heuristical rules to remove false plates: aspect ratio, maximum and minimum size... In our tests, on more than 50 cars, we always found the plate. And, only once, we also found false zones. Sometimes, the rectangle which encloses the V.L.P. is bigger than should be. This creates problems for the next stages.

2.2 Threshold Computation:

This is a method designed specifically for this problem. The base of the method is that we have computed the gradient of the image at the former stage. If we move along the image, the gradient will tell us where there are changes from foreground to background color (in fact, we located the characters on this basis).

Algorithm: choose a line of the image where the probability of having many transitions is high (for example, at the center). Then move along the line and detect all the transitions. Each transition represents a change in the state of a variable with only two possible values. We will call this variable Q, and its values will be 0 and 1. When there are no transitions we will be at a pixel of gray level L. Then we will compute:

$$Sum_Q = Sum_Q + L \qquad \text{(the sum of values for each state).}$$

$$Number_Q = Number_Q + 1 \qquad \text{(the number of pixels for each state).}$$

After the whole line we will compute:

$$Mean_Q = Sum_Q / Number_Q \qquad \text{(the mean for each state).}$$

The threshold is:

$$U = (Mean_0 Number_1 + Mean_1 Number_0)/(Number_0 + Number_1)$$

This method can be extended to several lines. The basis of the algorithm is the division of pixels between two states; representing black and white, of course. The problem is that, in a particular line, we don't know which state corresponds to each color. We solve that problem comparing the means of the states (the smaller is the black one). Then, we can compute a global sum for the white pixels and another for the black ones. We also count the pixels in each category and the final threshold computation is made using the same formula.

We apply this extended algorithm to the 80% of the lines in the plate (we discard the first and last 10% lines). This is approximately the character zone.

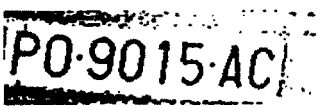

Fig. 2. Binarization with this method of the plate found for the car in Figure 1.

In the V.L.P. case, this method outperforms the classical algorithm of [2]. With this method characters don't get linked to the borders or to each other. A problem would be derived from thinner characters (harder to recognize). We designed a contour softening filter to apply after the segmentation that avoids that problem.

2.3 Skew Angle Correction:

The key fact, here, is the computation of the skew angle. This is achieved using the Hough transform. The algorithm is described in [3]:

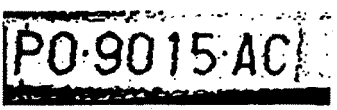

Fig. 3. Image of Figure 3 corrected with this method (skew angle = 3.3°).

2.4 Character Segmentation:
Segmentation into Rows:

To segment the plate into rows (after an stage that removes borders), we compute the vertical projection of the characters and divide the plate by the minimal of this projection. We only have to distinguish between two cases: one row and two rows.

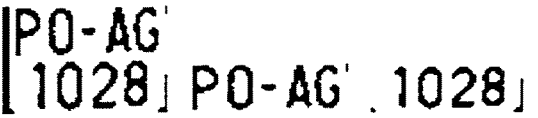

Fig. 4. Separation of rows for a square plate.

Segmentation of Characters:

To segment the characters we make an adaptive search of the connected black spots (if characters were white we would have made a negative of the image). We start the search at the center, and then move to the left and to the right. When we find a character we adapt our searching position to its coordinates. We apply many heuristical rules to remove spots that are not characters: aspect ratio, relative size... We allow a maximum length of search path with no character found, after that length the search would be stopped.

After the segmentation process, the characters are filtered through a simple method designed for contour softening. Each black pixel in the input will be also in the output. For a white pixel, we count the black and white pixels surrounding it. If black neighbors are majority the pixel will be turned into black. We can call it a modified majority filter.

As an example, we can see the results for the image in figure 5:

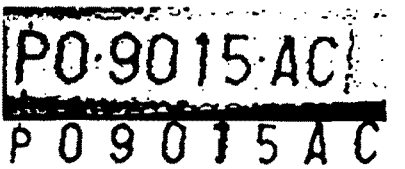

Fig. 5. Segmentation of characters and softening filter.

2.5 The Final O.C.R.:

Our final O.C.R. is based on the comparison of the actual character, extracted from the input image, with an alphabet containing all of the possible characters. We extract some characteristics from the images and we define a "metric" (or "distance") between them (a "template-matching" approach).

We decided to extract the horizontal and vertical projections of the bi-level image. Id est: the sum of the columns and of the rows. A background pixel counts as 0 and a foreground one counts as 1. We have to normalize, dividing by the size of the column/row we have just added (to make the projection be size independent).

We also compute what we call the number of strokes. This is defined as the number of foreground pixels zones (black strokes) we pass through when computing the projection. This will be necessary for the computation of the distance.

To define the metric or distance we considered that we would have to compare different size projections. We defined a heuristic comparison algorithm [4]. This algorithm performs a comparison using some overlap. We made the assumption that the alphabet image is always the biggest (if not, we decimate the input image). We compare

each value of the smaller vector with one or more values of the bigger (the number of values is equal to the quotient between projection lengths). We apply a special penalty when the number of strokes of the input projection is different from that of the alphabet. The final distance between the two characters is the sum of the absolute values of the distances between horizontal and vertical projections.

We could say that the number of strokes divides the projection curve in certain significant zones. And we must compare only corresponding zones (if they exist).

Our O.C.R. simply computes the distance of the input character to all the possible characters and gives as solution the one with the minimum distance. To reduce the number of comparisons we compute the number of holes in the input character and we only make the comparison with those characters in the alphabet with the same number of holes.

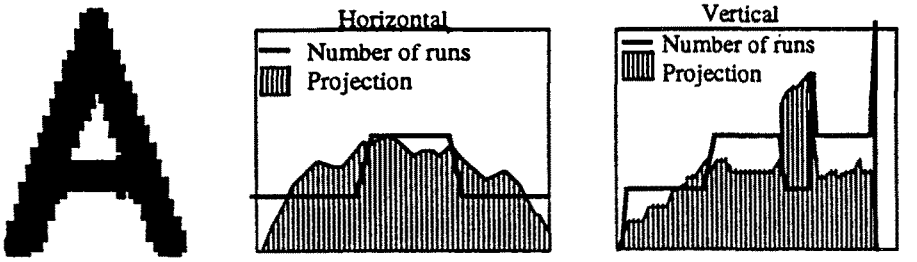

Fig. 6. Projections and number of strokes for the character "A".

We can see some noise peaks in the number of strokes. These are due to isolated pixels in the contour of the character. The problem can be overcome with the cleaning filter in the segmentation process. Another interesting protection against the isolated pixels is to establish a minimum length for each stroke to be taken into account.

The O.C.R. has (when the segmentation has worked perfectly) an accuracy of 91.0%. Most errors are on "difficult pairs" (8 and B, D and O ...) when the input character contours are distorted by noise.

3 Image Acquisition:

3.1 Camera and Lightning Considerations:

In the practical tests of this system we learned about the importance of a good camera election and configuration. It's also very important a good lightning. In this section, we will speak about this considerations.

We are interested in cars that are moving at a maximum speed of, say 36 Km/h. If we don't want the images to get blurred we need to use a camera with adjustable electronic shutter. A car at 36 Km/h moves 1 cm in one millisecond. This was set empirically to be our working limit. This obliges to take the image in one millisecond => a shutter of 1/1000.

We must set the lightning conditions to avoid an illumination gradient over the plate (this would invalidate threshold computation). Another thing to avoid is the direct

reflection of the focus light on the plate that hits directly the camera. This practically erases a piece of the plate. An ideal focus and camera configuration would be:

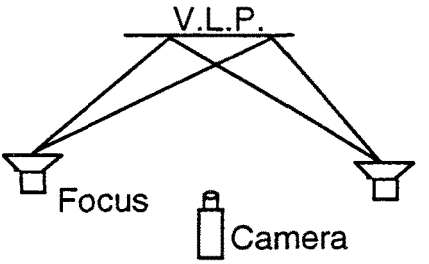

Fig. 7. Ideal configuration of camera and lights.

The superposition of light from two focuses avoids gradients. And the focus positions are such that the reflections never go onto the camera.

3.2 Application of Multirresolution Techniques:

We capture the images at a resolution of 768x576 pixels (divided into two fields due to the interlaced nature of the TV signal). This is the maximum resolution achievable for a PAL signal. Although this image is suitable for the segmentation and final O.C.R., it's too large for the first stage. This stage has to deal with the whole image, and working at full resolution would have a very big real-time penalty. There are also implementation problems in dealing with such a big block of memory.

We decided to use a decimated image for the first stage where the possible plates are located in the image. Once, the corresponding rectangular sub-images have been cut, we will work at full resolution from that point on.

Another problem arises when the car is moving. As it was explained in the former point we use a strong shutter to avoid blurred images. This means that each field of the standard TV PAL signal is sampled in a very short time. The camera gives us an "interlaced signal" where the odd and even fields are not equal, like the car in figure 8.

Fig. 8. Blurred image of a moving car.

This horribly distorted image is the actual input image for all the examples in this paper. We overcame the problem doing an "non-uniform sampling" retaining only even lines when we decimate (by a factor of 3).

Another correction is necessary when we cut the rectangles that represent the possible plates. Again, we only sample the even lines. The odd ones are computed via linear interpolation.

4 Experimental Results:

4.1 Practical Achievements:

Accuracy Considerations:

Depending on the illumination conditions, the accuracy varies considerably. The system is now installed in a parking with very different illumination conditions for the cameras at the input and at the output and the results are best for the camera with better illumination conditions. To make an average we picked out more than 50 images from the two cameras and obtained the following results:

Total number of cars:	51
Exact Result:	33
One False Character:	0
One Character Lost:	1
One Character Mistaken:	9
More than One Error:	8
No Result (no V.L.P. found):	0

We have found that the main errors come from the segmentation process.

Real-Time Considerations:

The mean time of execution of the whole process on a single image was of 0.95 seconds on a pentium PC at 100 MHz. We can consider this a quite good speed. The practical system applies the algorithms in the background whenever possible making computing time less critical for real-time performance.

4.2 The Parking Practical Application:

At the time of writing, this system is installed in a parking. The system is being used for monitoring and accounting. It controls who and when gets in and out. Also computes the time a car has been in and gives the amount of the bill.

4.3 Comparison with Other Similar Projects:

We only found a scientific publication that described a similar system [5]. This system applies a global threshold to the whole image and finds the plate using region growing techniques. Afterwards the V.L.P. is recognized via syntactic methods. The results are similar to ours.

We also know about various commercial systems to recognize V.L.P.'s. We only had access to user specifications, not to their technical foundations. We can say our system is at the same level.

5 Future Lines:

We consider this system still of low accuracy and so, we have many ideas to test in the future. A little summary could be:

- The application of vertical gradients to improve the accuracy of character location.
- To improve the final O.C.R. We would need a method able to decide when the input is not a valid character. This would permit the introduction of a more intelligent (feedbacked by the O.C.R. results) segmentation method.

6 Conclusions:

We have designed a system able to recognize the license number of a car having as input an image of it. We made no assumptions about size, color, or nationality of the V.L.P.

The algorithms used are easy and simple, although we have to experiment new ones to improve the results. In fact, the system is still under development.

This system has many practical applications. As examples of applications we can mention: parking accounting systems (current application), traffic monitoring and security systems.

7 References:

[1] J. Ohya, A. Shio, S. Akamatsu. "Recognizing Characters in Scene Images". IEEE Transactions of Pattern Analysis and Machine Intelligence. February 1994.

[2] N. Otsu. "A Threshold Selection Method for Gray Level Histograms". IEEE Transactions on System, Man and Cybernetics. January 1979.

[3] G. S. D. Farrow, M. A. Ireton, C. S. Xydeas. "Detecting the Skew Angle in Document Images". Signal Processing: Image Communication. 1994.

[4] J. L. Fernández, X. Fernández. "Un Nuevo Método para el Reconocimiento Automático de Matrículas". Proceedings of URSI-95, Valladolid (Spain), 1995.

[5] J. R. Cowell. "Syntactic Pattern Recognizer for Vehicle Identification Numbers". Image & Vision Computing. February 1995.