Peer-reviewed version. F. Martin, E. Vazquez-Fernandez, A. Formella, H. Gonzalez-Jorge and A. Dacal-Nieto, "Automatic reading of digital instrumentation," 2008 IEEE International Symposium on Industrial Electronics, Cambridge, UK, 2008, pp. 913-918, doi: 10.1109/ISIE.2008.4677066.

Automatic Reading of Digital Instrumentation

Fernando Martín[1], Esteban Vázquez Fernández[1,3], Arno Formella[2],
Higinio González Jorge[3], Ángel Dacal Nieto[3].

[1]Communications and Signal Theory Department. University of Vigo.
[2]Computer Science Department. University of Vigo.
[3]Laboratorio Oficial de Metroloxía de Galicia.
fmartin@tsc.uvigo.es, formella@ei.uvigo.es, evazquez@lomg.net

***Abstract*- This communication describes a computer vision system designed to automatically read the displays of digital instrumentation. The system is used in calibration sessions where many measurements have to be made and where we are interested in getting the whole series downloaded on a host computer. Before our system was running, a human operator had to inspect the instruments at the right times required by the calibration protocol and write down all the results. Note that we are speaking of very simple and sometimes old instruments that usually do not provide a digital interface or a removable memory.**

I. Introduction

A. Purposes of Development

The computer vision system as described in this article was designed to automatically read digital instrumentation measurements avoiding a boring and sometimes frustrating work of human operators. The system was first implemented at the "Laboratorio Oficial de Metroloxía de Galicia" (LOMG: www.lomg.es) for the task of digital thermometer calibration (Figure 1). Nevertheless, our application will be useful with all types of instruments exhibiting a numerical display.

As we will see later, we start with a photograph of the instrument displaying a stable measurement. Then we use standard image processing techniques to segment the image characters in a manually selected region of interest. Finally, we will see how we recognize the digits with our new approach that combines two different classifiers.

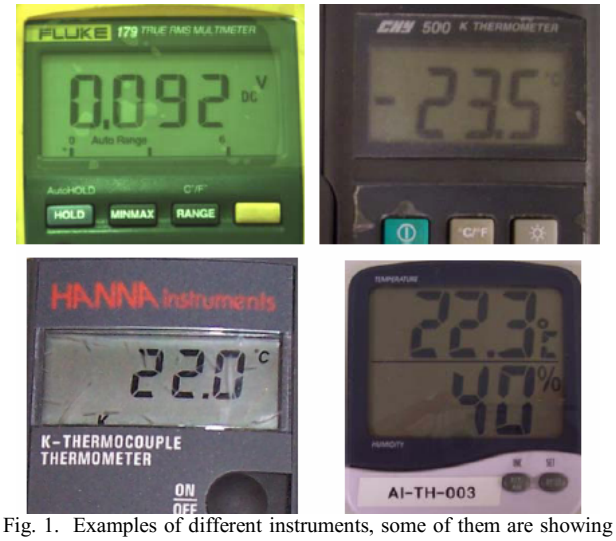

Fig. 1. Examples of different instruments, some of them are showing display defects (bubbles in the first one, stripes in the third one).

Fig. 2. Examples of different displays using different character fonts. Our system is able to deal with all of them.

B. Related Work

Our system is an example of character recognition in scene images like the one implemented in [1] in a more general context. Notice that we are not locating the character region (like [1] does) but we have taken some ideas from there like the interpolated threshold in non uniformly illuminated images.

Our system also shares ideas with car plate recognition systems, e.g. [2], [3], [4]. In such a plate recognition system designers cannot rely on human help to locate the plate but conversely a fixed character font is assumed. In contrast, for digital instrumentation, we have to take into account multiple fonts (Figure 2) which is, as we will see, a source of problems. Locating the regions of interest automatically is not an issue of high priority, because the operator must set up the calibrating experiment anyway.

The next section describes the methods we used for preprocessing (e.g., binarization or skew angle correction) and segmentation which are standard image processing techniques adequately adjusted to our problem.

The recognition step, as detailed in Section III, combines two methods: the first one is a classical one based on feature extraction followed by distance classification; the second one is an original method especially suited to recognize instrumentation digits (as we will see is somewhat inspired by the classical 7-segment display). The fusion of both recognizers is also an original contribution where we use the second recognizer to correct possible errors of the first one.

II. Image Capturing and Preprocessing

A. Image Capturing

Capturing is perhaps the most important part of the whole system. A good capture will make recognition easy while a bad one would make it impossible.

Due to the fact that we can not alter some laboratory conditions at LOMG, image capturing has to be done without modifying the environment illumination. We use a C-Cam BCi4 camera with 1280x1024 resolution (Figure 3). In most of the cases, the operator can use a 25 mm lens that is able to focus from 15 cm to 1 m. However, sometimes the physical conditions oblige the use of a 75 mm lens with a focal distance from 1.5 m to 10 m.

Fig. 3. Camera used for capturing.

As the observed instrument is not moving during image capturing, we decided to rely on the user to extract the regions of interest (Figure 4). The regions will have to be marked only once for all the series. Currently, we are working to achieve an automatic detection, using the methods described in [4] and, perhaps, taking advantage of the sequence of images.

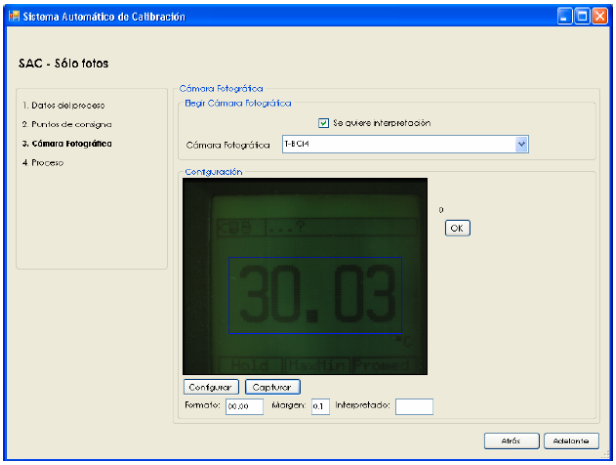

Fig. 4. User selection of the region of interest.

B. *Binarization*

Binarization is the process that converts a grayscale or color image into a binary one with only two levels.

We start by converting our colored images to grayscale giving preference to the ITU-R BT.709 recommendation (gray = 0.2125R + 0.7154G + 0.0721B), because instrument displays often are green or, at least, dominated by the green channel. As expected, the resulting gray level distribution shows a bimodal histogram (two main peaks, see Figure 5).

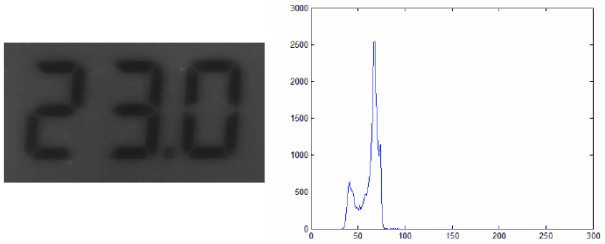

Fig. 5. Left: example of a grayscale display image. Right: gray histogram of the left image.

We use a combination of the well known Otsu method from [5], implementing it via an approximate iterative version found in [6] and the method taken from [7] that is based on searching the histogram peaks (and locating thresholds on the minima between them).

Fig. 6. Left: Peak detection method. Right: iterative Otsu method.

As can be seen in Figure 6, the Otsu method can create some segmentation problems due to the thicker characters it produces. We also experienced some problems with images with important illumination gradients (Figure 7). In these cases, a global threshold is not enough. This can be solved dividing the image into sub-images and applying interpolated thresholds [1] (the improved results are shown in Figure 8).

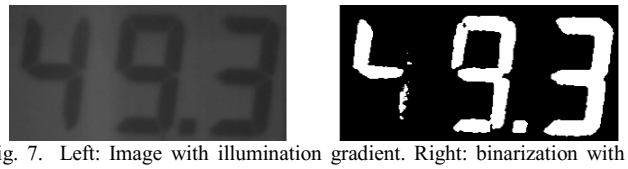

Fig. 7. Left: Image with illumination gradient. Right: binarization with a single threshold.

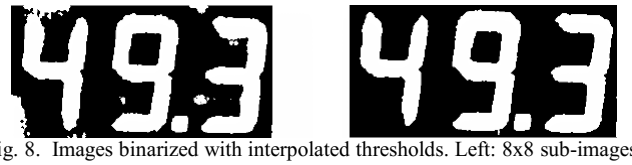

Fig. 8. Images binarized with interpolated thresholds. Left: 8x8 sub-images. Right: 4x3 sub-images.

The final solution consists of applying first the peak detection method and then measuring threshold quality using the histogram area in the threshold neighborhood. If that area is bigger than usual the threshold is considered incorrect and we switch to an interpolated threshold with 12 sub-images (4x3 sub-image grid). We use Otsu threshold on each piece.

To measure the threshold quality, we compute the histogram area in the threshold boundary (Figure 9). The area should be

small, so if it is exceeds 2.5% of the entire histogram area we will consider it as an invalid threshold.

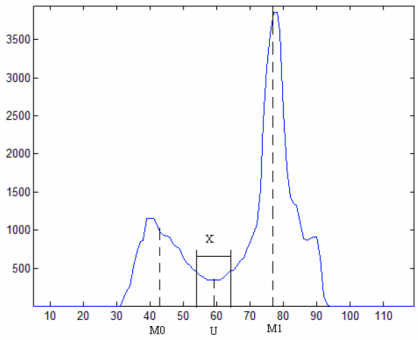

Fig. 9. Threshold quality measuring, X=0.15(M1-M0).

C. *Skew Angle Correction*

To correct a possible skew angle, we estimate the upper contour of the characters and compute the slope of the resulting straight line (see Figure 10).

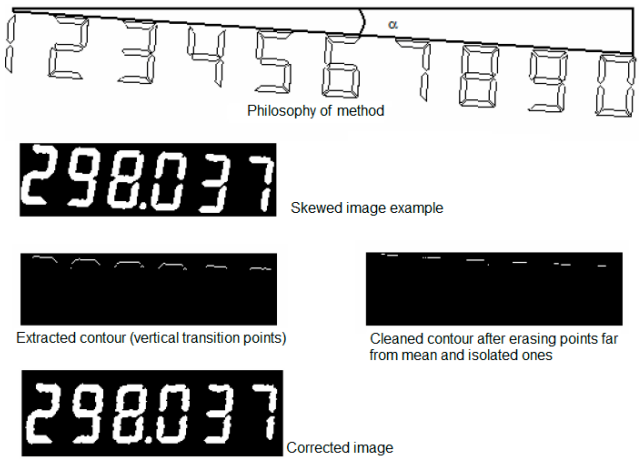

Fig. 10. Skew angle correction.

D. *Character Row Extraction*

Extracting the character row is the same as removing the blank lines above and below the characters and also to the left and to the right of them. As shown in Figure 11, this is an easy process using an horizontal and a vertical image projection.

On each projection, we detect the region of interest beginning and ending by searching for large gradients, first from left to right (top to bottom) and afterwards in the opposite direction.

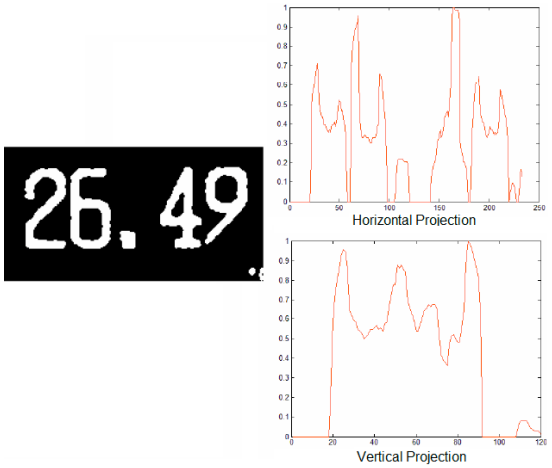

Fig. 11. Binary display image and both projections.

III. CHARACTER SEGMENTATION

To isolate the different characters in a preprocessed row, we use what we call "enhanced projections". For instance, the enhanced horizontal projection of an image is the vector that contains in position i the dot product $(< x, y >= \sum x_j y_j)$ between the $(i-1)^{th}$ and $(i+1)^{th}$ column. Figure 12 illustrates the advantages of an enhanced projection. As can be observed, the minima that mark character transitions are deeper compared to the standard projections.

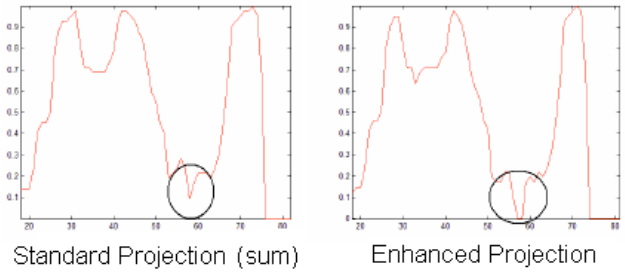

Fig. 12. Standard projection (left) and enhanced one (right).

The main procedure consists of searching the horizontal projection from right to left while applying a kind of hysteresis process. The right to left direction is chosen as the digit ending is usually more evident than its beginning (most digits end by a vertical line, i.e., a strong projection gradient). We use the word hysteresis because the threshold to detect a character beginning is different (bigger) to the threshold used to detect an ending (Figure 13).

To detect segmentation errors (linked characters), we compute the aspect ratio of all segmented items (R=height/width). For a ratio R of less than 1.2 we decide that we rather have two characters. In this case, we first compute the local maxima and minima of the projection (using a sliding window procedure as described in [7]). The deeper minimum

that is between two peaks is selected as the optimum breaking point.

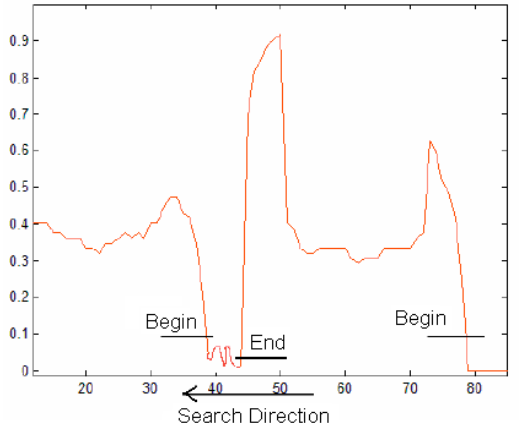

Fig. 13. Detection with hysteresis.

In most cases, our segmentation is able to detect the decimal point (something very important for this application). Sometimes, however, we have skewed characters that confuse the projection. When the point is not found in the first run, we try again using a projection of the lower part of the characters, note that characters are already segmented (Figure 14).

Fig. 14. Above: skewed characters that make the point detection difficult. Below left: lower part of "2". Below right: projection of fragment.

Even with this effort, the decimal point does not yield a correct detection rate bigger than 70%. That is why we ask the user for the position of this point (number format) before beginning recognition of a series of images.

IV. Digit Recognition

First, we normalize the extracted characters by scaling them to a fixed size of 16x16 points. Note that we do not maintain the aspect ratio. We had started by keeping that ratio (getting a character with 16 points height and less than 16 points width) followed by centering with vertical lines [8]. Nevertheless, we discovered that, in the end, we yielded slightly worse results. Distorting characters is not a problem as long as we also distort the patterns as well.

A. Feature Extraction

We extract features in two ways. First, we take advantage of the already computed horizontal and vertical projections of the individual characters. Obviously, different characters may have almost the same projection (see Figure 15). Therefore, we split each character into two halves (an upper and a lower one) and then compute 4 vectors: upper horizontal (16 values), upper vertical (8 values), lower horizontal (16 values), lower vertical (8 values). So projections yield a feature vector of length 48.

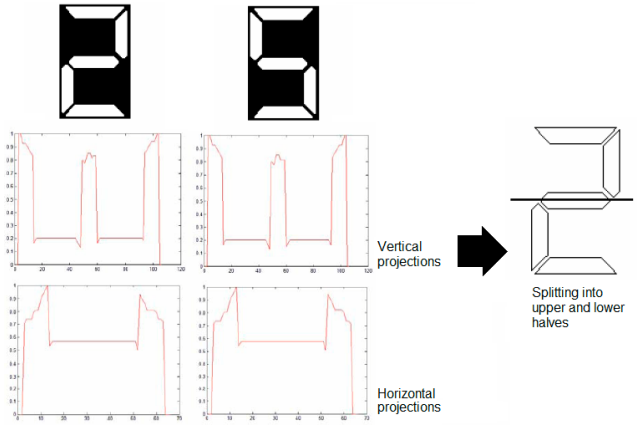

Fig. 15. Two characters with almost identical projections.

Second, we use features based on Kirsch gradients [6], [8]. The Kirsch operator computes a first order derivative (similar to other operators like Prewitt, Sobel, Canny, etc. [9]). Our purpose is to compute image components along four directional axes: horizontal, vertical, right diagonal and left diagonal. For example, the horizontal component is computed via a vertical gradient (being always perpendicular to the desired direction).

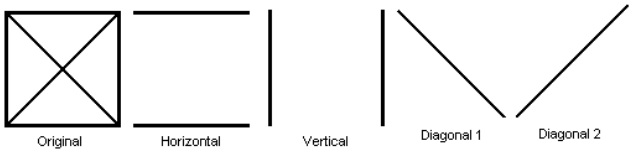

Fig. 16. Directional components.

Kirsch defines an algorithm that uses the following notation to index the pixels as neighbors to a certain pixel (i,j):

A_0	A_1	A_2
A_7	(i, j)	A_3
A_6	A_5	A_4

The equations are:

$$S_k = A_k + A_{k+1} + A_{k+2}$$
$$T_k = A_{k+3} + A_{k+4} + A_{k+5} + A_{k+6} + A_{k+7}$$
$$G_D(i,j) = \max\left\{1, |5S_D - 3T_D|, |5S_{D+4} - 3T_{D+4}|\right\} \quad (1)$$

Where $G_D(i,j)$ is the gradient for pixel (i,j) in direction D and sub-indexes of A are evaluated modulo 8.

With the help of these equations, we compute feature maps in four directions: horizontal (H, D=0), vertical (V, D=2), right diagonal (R, D=1) and left diagonal (L, D=3).

Eventually, we obtain four local feature maps as 16x16 images. Using all of them as a feature vector would result in a vector of length 1024. In [8] it is suggested to decimate the 16x16 images to size 4x4. However, in our particular system we obtained better results leaving the 4 directional components in its original size (Figure 17).

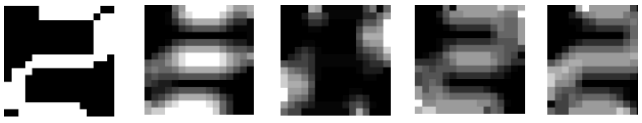

Fig. 17. True example of directional components.

We combine the two feature vectors, i.e., projections and Kirsch, yielding a total vector of length 1024+48=1072.

The relative sizes of the two vectors to be joined are so different that perhaps the conclusion may be drawn that the projection vector is not important. Note that we were planning 4x4 components (a final vector of length 64+48).

B. Classifier

We tried various classifiers (like probabilistic neural networks, Gaussian classifiers and k-NN) and achieved the best results for the nearest neighbor algorithm (1-NN). This is not very surprising as it is explained in [10]. In our system, we have to deal with several different character types (7-segment, graphical fonts, skewed, not skewed, etc.). In this multi-font situation, there exists sometimes more variance between the samples of the same character in different fonts than between different characters in the same font (intra-class variance greater than inter-class variance, Figure 18).

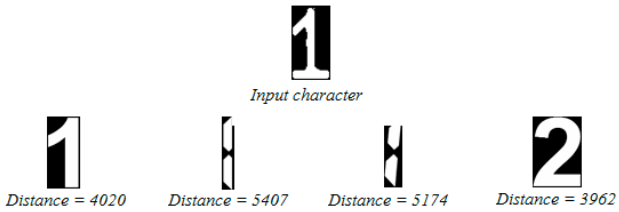

Fig. 18. Example of inter-class and intra-class distance. Input character '1' is closer to a '2' than to other '1's of different fonts.

As patterns for the 1-NN classifier we chose perfect ones (obtained from the different fonts). We tried to use patterns from segmented input digits but the 1-NN got better results for the artificial, perfect ones.

C. Classification via Visual Inspection

Visual inspection is an intuitive method. We developed it studying the reasoning that people express when they describe how they recognize characters.

The alphabet of digits ('0' to '9') in different fonts is reduced and we can find a defining characteristic for each class. For example, no matter which font, the number **'2'** has always two openings: one in the upper left part and the other in the lower right one.

We implemented this method as a complete recognizer based on a template that defines the regions of interest (Figure 19). We check whether the seven regions are active or not by majority voting between foreground and background pixels. Note that the template is a 7-segment detector where we have removed the corners (which are sharp in some fonts and rounded in others).

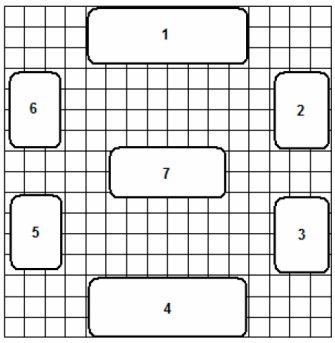

Fig. 19. Regions of interest tested (a 7-segment like template).

Visual inspection by itself does not achieve a high enough recognition rate but combined with the more classical approach described above we got a very robust system.

D. Classifier Fusion

To benefit from both classification schemes, we run them in parallel and combine their results in the following manner (see Figure 20 for a flow diagram):

We run the feature extraction and compute the norm-1 distance to every pattern yielding a distance vector.

We run the visual inspection algorithm yielding an estimate for the digit to be recognized. This is coded as a binary vector of length 10 where an active bit at a position corresponds to recognition of that class.

We reduce the distances that correspond to the class that was recognized by visual inspection by 20%, empirical.

We apply 1-NN and minimum distance wins.

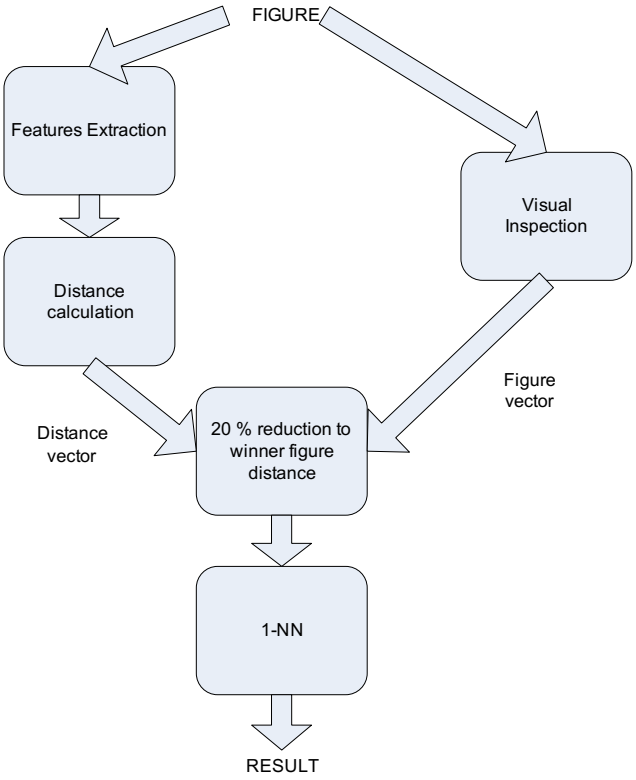

Fig. 20. Flow diagram of classifier fusion.

V. RESULTS

Our test set consisted of 16 image sequences, with a total of 449 images (examples are shown in Figure 21). The system obtained the correct values 439 times, i.e., 98% recognition rate (measured on display images, not on individual digits). We have tried samples from all accessible fonts: 7-segment, skewed, not skewed, graphic display, etc. In routine work of the LOMG, the 7-segment displays are the most common ones, resulting in an even higher recognition rate for the real distribution of display types.

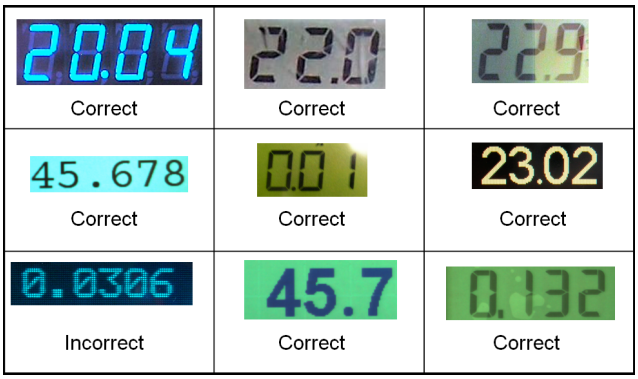

Fig. 21. Some examples from our test set.

Fig. 22. Image with erroneous result.

As an example, see that the erroneous image in figure 21 was due to the special nature of this display. This machine uses dotted characters that we are not able to segment (Figure 22). Perhaps, we could solve this using mathematical morphology (closing [6]) before segmenting. But we should pay attention to avoid spoiling good results on other images.

VI. CONCLUSION AND FUTURE LINES

We have designed and implemented a useful system able to read almost any display of digital instrumentation devices.

We have employed standard image processing techniques adapted to this problem. We also designed a hybrid recognizer, which combines a classical classifier with a visual inspection algorithm.

The recognition rate of the final system suggests that we have solved the problem despite the intra-class variance due to the presence of multiple fonts.

As future work lines, we emphasize on the following:

Designing an automatic location algorithm to avoid that users have to define the region of interest.

Optimizing the feature vector trying to detect the principal components.

Using knowledge from previous images (for instance the font type) when recognizing subsequent images in a sequence.

Detecting automatically displays that are not stable yet (i.e., its digits are changing from one value to another).

REFERENCES

[1] J. Ohya, A. Shio, S. Akamatsu. "Recognizing Characters in Scene Images". IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol. 16, Nº 2, pp 214-220. February, 1994.

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

[3] X. Fernández Hermida et al. "Automatic and Real Time Recognition of V.L.P.'s (Vehicle License Plates)". Lecture Notes on Computer Science. Springer-Verlag. Nº 1311, Vol 2, pp 552-559. 1997.

[4] F. Martín, M. García. "New Methods for Automatic Reading of VLP's (Vehicle License Plates)". Proceedings of SPPRA-2002 (Signal Processing Pattern Recognition and Applications). June, 2002.

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

[6] R.C. González, R.E. Woods. "Digital Image Processing". Ed. Prentice Hall, 2º edition, 2002.

[7] F. Martín. "Analysis Tools for Gray Level Histograms". Proceedings of SPPRA-2003. June, 2003.

[8] D. Cruces, F. Martín. "Printed and Handwritten Digits Recognition Using Neural Networks". Proceedings of ICSPAT-97 (International Conference on Signal Processing and Applications Technology). September, 1998.

[9] A. K. Jain. "Fundamentals of Digital Image Processing". Ed. Prentice Hall. 1989.

[10] J. L. Blue et al. "Evaluation of Pattern Classifiers for Fingerprint and OCR Applications". Pattern Recognition. Pergamon Press. Vol 27, Nº 4, pp 485-501. 1994.