Road Traffic Sign Detection and Classification

Arturo de la Escalera, *Member, IEEE,* Luis E. Moreno, *Member, IEEE,*
Miguel Angel Salichs, *Member, IEEE,* and José María Armingol

***Abstract*— A vision-based vehicle guidance system for road vehicles can have three main roles: 1) road detection; 2) obstacle detection; and 3) sign recognition. The first two have been studied for many years and with many good results, but traffic sign recognition is a less-studied field. Traffic signs provide drivers with very valuable information about the road, in order to make driving safer and easier. We think that traffic signs must play the same role for autonomous vehicles. They are designed to be easily recognized by human drivers mainly because their color and shapes are very different from natural environments. The algorithm described in this paper takes advantage of these features. It has two main parts. The first one, for the detection, uses color thresholding to segment the image and shape analysis to detect the signs. The second one, for the classification, uses a neural network. Some results from natural scenes are shown. On the other hand, the algorithm is valid to detect other kinds of marks that would tell the mobile robot to perform some task at that place.**

***Index Terms*—Advanced driver information systems, color/shape processing, computer vision, neural networks, traffic signs recognition.**

I. Introduction

COMPUTER vision has three tasks in order to make a road vehicle fully autonomous. Under the "road detection and following" titles are those algorithms that allow a vehicle to drive on roads and highways. They have been studied for many years and with many good results. Different approaches have been used: color segmentation, control theory, neural networks, etc. [1]–[5]. Obstacle detection (and avoidance) is an open research area, where there have been a large number of contributions since the 1980's. Automatic recognition of traffic sign studies started more recently, but are increasing rapidly. A system capable of performing such a task would be very valuable and would have different applications. It could be used as an assistant for drivers, alerting them about the presence of some specific sign (e.g., a predefined exit on a highway) or some risky situation (e.g., driving at a higher speed than the maximum speed allowed). In autonomous vehicles, traffic signs should provide the control system with similar (and also some specific) information to that offered to human drivers. It is also possible to design specific signs, for mobile robots, with a format similar to traffic signs, that would indicate some kind of information about tasks, prohibitions, or warnings in the environment.

Luo *et al.* [6] carried out many studies in this area. They have detected the sign by a fractal texture segmentation and have used two different neural networks. A receptive field neural network [7], with an input layer of 32×32 neurons, an output of ten neurons and four hidden layers of 16×16, 8×8, 4×4, and 30 neurons, where the net was trained to recognize nine traffic signs. For each sign, images at three different distances (1, 2, and 3 m) were chosen. Although it is 99% certain, the recognition time is 4 s, which is quite long for a real-time case. The second type is a reconfigurable neural network [8], with an input layer of 45×45 neurons, ten outputs, and the recognition time is 1 s. Blancard [9] recognized the signs by their color and form. There are three classified types: 1) octagonal signs; 2) warning signs; and 3) the "stop" sign. For the color classification, a passband filter for the chosen color (red) is attached to a black and white charge-coupled device (CCD) camera. Then, a Sobel filter is applied, and the edges are found in the image by connecting a pixel with its neighbors by the Freeman code. Some features are calculated from the resulting contours: perimeter, length, gravity center, and compactness, and Freeman code. These features are the inputs to a neural-network-type restricted coulomb energy (RCE) for the classification. Within each sign type, the algorithm does not detect what sign it is, in particular; on the other hand, the detection is at a fixed distance from the vehicle. The detection time is 0.7 s. Piccioli *et al.* [10] used black and white images. After the extraction of the edges, there is a shape analysis looking for circular and triangular contours. When they are found, the subimages are normalized to 50×50 pixels, and the classification is done through a cross correlation with a database. Much research has been carried out under the PROMETHEUS (PROgraM for European Traffic with Highest Efficiency and Unprecedented Safety) project. Besserer *et al.* [11] created a pyramidal structure from the original image. An edge detector is applied to every image of the pyramid. The edges of an image joined the edges of the upper image. By analyzing the generated contours, the signs are classified into triangular, circular, or rectangular signs. The inner part of the sign is not analyzed. Estable *et al.* [12] developed the fastest system (200 ms). Their hardware consists of four PowerPC's (601) and four transputers (T805). There is a color clasiffication through a neural network, a shape analysis of the region contours previously detected, and a pictogram classification through a radial-basis-function neural network.

Manuscript received July 23, 1996; revised September 3, 1997. This work was supported by the Spanish Government under CICYT Project TAP94-0711-C03-02.

The authors are with the Area de Ingenieria de Sistemas y Automatica, Universidad Carlos III de Madrid, 28911 Madrid, Spain (e-mail: escalera@ing.uc3m.es).

Publisher Item Identifier S 0278-0046(97)08490-6.

The algorithm presented here has two steps. The first one localizes the sign in the image depending on the color and the form. The second one recognizes the sign through a neural network.

II. Traffic Sign Detection

There are four types of traffic signs that are shown in the traffic code: 1) warning; 2) prohibition; 3) obligation; and 4) informative. Depending on the form and the color, the warning signs are equilateral triangles with one vertex upwards. They have a white background and are surrounded by a red border. Prohibition signs are circles with a white or blue background and a red border. Both warning signs and prohibition signs have a yellow background if they are located in an area where there are public works. To indicate obligation, the signs are circles with a blue background. Informative signs have the same color. Finally, there are two exceptions: 1) the yield sign, an inverted triangle; and 2) the stop sign, a hexagon. They were not studied here. To detect the position of the sign in the image, we must know the two properties we talked about before, i.e., color and shape.

A. Color Thresholding

The most intuitive color space is the RGB system. The color of every pixel is defined by three components: red, green, and blue. Because of this, the color threshold has the following expression:

$$g(x,y)=k_1\begin{cases}R_a\le f_{\mathrm{r}}(x,y)\le R_b\\ G_a\le f_{\mathrm{g}}(x,y)\le G_b\\ B_a\le f_{\mathrm{b}}(x,y)\le B_b\end{cases}$$
$$g(x,y)=k_2\quad\text{in any other case}\qquad(1)$$

where $f_{\mathrm{r}}(x,y)$, $f_{\mathrm{g}}(x,y)$, and $f_{\mathrm{b}}(x,y)$ are, respectively, the functions that give the red, green, and blue levels of each point of the image. One of the greatest inconveniences of the previous color space is that it is very sensitive to lighting changes.

That is the reason why other color spaces are used in computer vision applications, especially the hue, saturation, intensity (HSI) system that is very invariant to lighting changes [13]. The problem with HSI is that its formulas are nonlinear, and the computational cost is prohibitive if special hardware is not used. That is why we have modified the approach suggested by Kamada and Yoshida [14], i.e., the color ratio between the intensity of the specified color and the sum of intensity of RGB. Instead, we have used the relation between the components. Thus, the thresholding is (assuming that the red component is chosen as a reference)

$$g(x,y)=k_1\begin{cases}R_a\le f_{\mathrm{r}}(x,y)\le R_b\\ TG_a\le\dfrac{f_{\mathrm{g}}(x,y)}{f_{\mathrm{r}}(x,y)}\le TG_b\\ TB_a\le\dfrac{f_{\mathrm{b}}(x,y)}{f_{\mathrm{r}}(x,y)}\le TB_b\end{cases}$$
$$g(x,y)=k_2\quad\text{in any other case.}\qquad(2)$$

Thus, in Fig. 1, for example, the signs have a red border. Because of that, the pixels searched have a high red value with respect to their green and blue values. This thresholding can be implemented in a 16-bit lookup table (LUT) and can be achieved in real time. One example of the thresholding can be observed in Fig. 1.

Fig. 1. Red color thresholding.

B. The Optimal Corner Detector

The methods developed for corner detection can be classified into two groups.

1) The first is corner detectors that work on the codification of the edge of an object. This kind of detector requires a previous division of the image in regions, the subsequent extraction of the edges of these regions, and, finally, their codification. The accomplishment of all these steps requires a long calculation time [15].
2) The second is corner detectors that work directly on the image. Within this group, several detectors based on the gradient of the image are found [16]–[18]. The changes in the intensity and direction of the gradient serve as criteria to consider a given point as a corner. Neither of these methods differentiates the kind of detected corner, a circumstance that, as we will see, is useful for the detection of a sign. Finally, within this group, the optimal corner detector is included [19]. As it will be shown below, it obtains the corners from the convolution

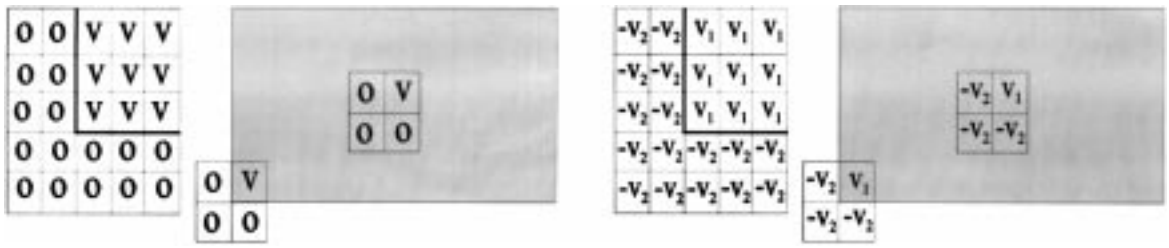

Fig. 2. Intuitive masks for 90° corner.

of the image with a mask. It is very appropriate for this application because of two characteristics, its speed and its capacity to decide the kind of detected corner depending on the mask used.

We said that the optimal detector will find the corners from the convolution of the image with a mask, but, what are these masks? They are correlation filters. The image of the corner of an object will be ideal if it has the same grey value for all the points of the object and zero in the background. Keeping in mind the definition of convolution and an ideal 90° corner, a mask that serves for its detection can be built in an intuitive way (Fig. 2). As every value of the mask is multiplied by the pixel value that is below, if the form of the corner is imitated within the mask, the convolution has the highest value for the corner by giving a positive value to the adequate cells and zero to the rest. This mask is not the ideal, since the same value is obtained in the corner as in the rest of the object. If now all the zeros are changed for a given negative value, when the mask is located on the object, its positive part contributes to increase the result of the convolution, but its negative part weighs more, and the value obtained is smaller. Since the value obtained for the background continues to be zero, a maximum in the corner is obtained and allows us to locate it. Though this latter mask operates much better that the previous one, we must consider that the image is real and not an ideal model. In a real image, the corner will not be so perfectly defined, and some noise will appear. To minimize the effects of this, one must seek some optimum values for the detection mask. It is precisely the obtainment of these values that characterizes the optimal corner detector.

To develop the detector, Rangarajan *et al.* [19] started building a mathematical model. Let us take, for example, a corner located in the origin so that the X axis will be the bisector of the angle formed by the two edges of the corner. Then, the function that describes the grey levels will be

$$I(x, y) = \begin{cases} A, & \text{if } x > 0 \text{ and } -mx < y < mx \\ 0, & \text{in any other case} \end{cases} \quad (3)$$

where m is the slope of the upper edge.

The model is completed taking into account the random variations that can be produced in the grey levels. If $n(x, y)$ is a white noise, the function that describes the grey levels around the corner is

$$F(x, y) = I(x, y) + n(x, y). \quad (4)$$

Once built, the model is used to find the sought function. This function, $g(x, y)$, has to fulfill the following criteria.

- The result of the convolution, $O(x, y) = F(x, y) * g(x, y)$, will be maximum in the corner.
- The operator $g(x, y)$ should not be sensitive to the noise.
- The operator $g(x, y)$ should not deallocate the corners; that is to say, the convolution $O(x, y)$ must be maximum for the point $O(x, y)$ in which this is located.

By expressing mathematically the previous criteria and by maximizing them, the optimal corner detector is obtained:

$$g(x, y) = c_1 \sin \frac{m\pi x}{W} \left[-\left(e^{zW} + e^{-zW}\right) + e^{yW} + e^{-yW}\right] \quad (5)$$

$$g(x, y) = c_2 \sin \frac{n_1\pi x}{W} \sin \frac{n_2\pi y}{W} \quad (6)$$

where W is the size of the mask, and c_1, c_2, m, n_1, n_2, and z are constants. Equation (5) corresponds to the clear part of the corner and (6) to the dark part. The constants c_1 and c_2 multiply all the values of the mask and, so, their value is arbitrary. We have chosen them as a scale factor, making c_1 equal to c_2. To decide the values of m, n_1, and n_2, let us assume that the grey levels of the object are higher than the ones corresponding to the background. Supposing the ideal case, in which there is no noise, it is clear that the mask that gives the maximum value in the corner is the one with positive values in the corresponding part of the object and negative values in the corresponding part of the background. The constant m appears in the first equation. It can be seen that the term is always negative, since $0 < y < W$, because it is the part corresponding to the object (remember that W is the size of the mask). When $m = -1$ and x varies from 0 to W, the term takes only negative values, and the obtained values from the first equation are positive. If m takes any other value, negative values will appear in the part of the mask corresponding to the object. Therefore, the value of m must be -1. To obtain negative values in the part of the mask that corresponds to the background (values that came from the second equation), one must get different signs for the terms. The constant should, therefore, be n_1 equal to 1 and n_2 equal to -1. The authors of the method have not found any rule to give a value to z, although, experimentally, they have noticed that an increase in the value of z increases the capacity for filtering the noise. The value chosen by them is z equal to 0.2.

The mask, which follows as an example, is created for 60° angles and has a size of 9×9 pixels. The bold numbers form the corner shape.

3	-5	-5	-3	0	-3	-5	-5	-3
-5	-9	-9	-5	0	-5	-9	-9	-5
-5	-9	-9	-5	0	-5	-9	8	5
-3	-5	-5	-3	0	6	9	9	6
0	0	0	0	0	6	10	10	6
-3	-5	-5	-3	0	6	9	9	6
-5	-9	-9	-5	0	-5	-9	8	5
-5	-9	-9	-5	0	-5	-9	-9	-5
-3	-5	-5	-3	0	-3	-5	-5	-3

The chosen size of the masks was 9 × 9, since, if they were smaller, they could not represent the corner correctly, and, if they were larger, they would affect a too-wide zone of the image and there would be problems in the case of the presence of some objects near the corner.

The triangular signs have three 60° corners, the corresponding detection masks of which are the following.

-6	-11	-11	-6	0	-6	-11	-11	-6
-11	-18	-18	-11	0	-11	-18	-18	-11
-11	-18	-18	-11	0	-11	-18	-18	-11
-6	-11	-11	-6	0	-6	-11	-11	-6
-6	-11	-11	12	12	12	-11	-11	-6
-11	-18	-18	19	20	19	-18	-18	-11
-11	-18	17	19	20	19	17	-18	-11
-6	-11	10	12	12	12	10	-11	-6
0	12	20	20	12	20	20	12	0

central upper corner, (60° type 1)

-6	-11	-11	-6	0	-6	7	7	4
-11	-18	-18	-11	0	-11	13	13	8
-11	-18	-18	-11	0	10	17	17	10
-6	-11	-11	-6	0	12	19	19	12
0	0	0	0	0	12	20	20	12
-6	-11	-11	-6	0	-6	-11	-11	-6
-11	-18	-18	-11	0	-11	-18	-18	-11
-11	-18	-18	-11	0	-11	-18	-18	-11
-6	-11	-11	-6	0	-6	-11	-11	-6

lower left corner, (60° type 2)

The mask for the lower right corner (type T3) is symmetrical with respect to a vertical axis of the mask for the lower left corner. To detect the corners of the "stop" sign (T4, T5, and T6 types), one has to use symmetrical masks with respect to a horizontal axis of the ones used for the warning signs. Then, there are six 9 × 9 masks for triangular signs. To reduce the number of masks, the 90° masks for the detection of the lower left and right corners can be used for the upper left and right corners of the triangular signs. If the masks are observed, we can see that the difference is not very large, and, since the grey level of the background is going to be low, the results, which follow, are very similar.

-6	-11	-11	-6	0	-6	7	7	4
-11	-18	-18	-11	0	-11	13	13	8
-11	-18	-18	-11	0	10	17	17	10
-6	-11	-11	-6	0	12	19	19	12
0	0	0	0	0	12	20	20	12
-6	-11	-11	-6	0	-6	-11	-11	-6
-11	-18	-18	-11	0	-11	-18	-18	-11
-11	-18	-18	-11	0	-11	-18	-18	-11
-6	-11	-11	-6	0	-6	-11	-11	-6

-6	-11	-11	-6	0	4	7	7	4
-11	-18	-18	-11	0	8	13	13	8
-11	-18	-18	-11	0	10	17	17	10
-6	-11	-11	-6	0	12	19	19	12
0	0	0	0	0	12	20	20	12
-6	-11	-11	-6	0	-6	-11	-11	-6
-11	-18	-18	-11	0	-11	-18	-18	-11
-11	-18	-18	-11	0	-11	-18	-18	-11
-6	-11	-11	-6	0	-6	-11	-11	-6

If the image processing board cannot perform 9 × 9 convolutions, the masks can be converted into a combination of smaller ones, as shown at the bottom of the next page.

As an example, the mask for a 90° corner can be decomposed in Sbm1, Sbm3, and Sbm4 masks (see Table I).

The result is

$$I_{\mathrm{Sbm}i} = I(x, y) * \ \mathrm{Sbm}i(x, y) \tag{7}$$

$$\begin{aligned} I(x, y) * M3(x, y) = {} & I_{\mathrm{Sbm1}}(x+3, y+2) \\ & + I_{\mathrm{Sbm4}}(x+3, y) \\ & + I_{\mathrm{Sbm2}}(x-2, y+2) \\ & + I_{\mathrm{Sbm2}}(x-2, y-3) \\ & + I_{\mathrm{Sbm2}}(x+3, y-3) \end{aligned}$$

where $I(x, y)$ is the function that describes the grey levels of the image.

C. Corner Extraction

To obtain a corner of an image, the algorithm follows these steps.

1) It obtains the convolution for every type of mask.
2) It selects the points above a threshold. This threshold is obtained from an ideal result. Thus, supposing that the corner is ideal, the result of the convolution will be maximum for this corner. The threshold is the necessary percentage to consider a point as a corner.
3) It calculates the center of mass. Although the detection mask is built to obtain the maximum value of the convolution exactly in the corner, because the image is never ideal and the threshold, a sole isolated point will never appear labeled as a corner. To prove this circumstance, the result obtained after applying the two first detection phases with 60° masks to an image is shown in Fig. 3. The center of mass is calculated by

$$x_{\mathrm{cm}} = \frac{\sum x_i}{n}, \qquad y_{\mathrm{cm}} = \frac{\sum y_i}{n}. \tag{8}$$

TABLE I
90° MASK DECOMPOSITION IN SMALLER ONES

Sbm3	Sbm1
	Sbm4
Sbm3	Sbm3

Fig. 3. Points detected as corners.

1) Triangular Sign Detection: Due to their shape, we can localize triangular signs by seeking in the image the three kinds of corners that form the triangle and by proving that they are, in fact, forming an equilateral triangle. The same principle can be applied to the "stop" sign, only, here, the triangle is downward. The steps for the detection of the triangular signs are as follows (Fig. 4).

1) The first is corner detection. All the corners that belong to the types that appear in the sign are sought in the image, i.e., upper vertex (T1 type), lower left vertex (T2 type), and lower right vertex (T3 type).
2) The second is the study of the position of the corners. A sign will be present when three corners are found forming an equilateral triangle.

To obtain that, the algorithm does the following.

1) The image is scanned until a T1-type corner is found.
2) From the T1 corner, a search zone is defined through two lines that start from it and have slopes of 52° and 68°. This area will also be limited by the maximum and the minimum heights that we expect the sign has in the image. In this area, a T2-type corner is sought. If it is not found, the algorithm returns to step 1, to continue scanning the image looking for another T1-type corner.
3) A second search area is created, delimited by two lines that start from the T1-type corner and have slopes of $-68°$ and $-52°$ ($\pm 6°$ has been considered the maximum rotation allowed to the signs), respectively, and two lines depending on the position of the T2-type corner:

$$\text{Height}_{\max} = \text{Height}_{\text{corner2}} + \frac{a}{2}$$
$$\text{Height}_{\min} = \text{Height}_{\text{corner2}} - \frac{a}{2} \tag{9}$$

where a is the width obtained in the second step.

In this second area, a T3-type corner is sought. If it is found, the algorithm considers that the three corners found in the successive stages correspond to a sign; otherwise, it returns to step 2.

The fact of differentiating in every step the kind of corner that is being sought is important to avoid mistakes. Some points on the edge of a sign may appear labeled as corners, although, in reality, they are not (Fig. 5). This is due to the mask size and the threshold chosen to consider a point as a corner. However, if the figure is observed, there are points labeled as the T2-type corners in black and those of the T3 type in white, and we can see that the wrong points do not affect the detection, since they appear in areas in which they are not sought. The result of applying the algorithm on a real image can be observed in Fig. 8(a).

2) Rectangular Signs Detection: A rectangular sign is formed by two horizontal and two vertical sides. The algorithm seeks the four kinds of 90° corners that form the sign and that are located defining a rectangle.

Sbm1

4	7	7	4	0
8	13	13	8	0
10	17	17	10	0
12	19	19	12	0
0	0	0	0	0

Sbm2

12	19	19	12	0
10	17	17	10	0
8	13	13	8	0
4	7	7	4	0
0	0	0	0	0

Sbm3

-6	-11	-11	-6	0
-11	-18	-18	-11	0
-11	-18	-18	-11	0
-6	-11	-11	-6	0
0	0	0	0	0

Sbm4

12	20	20	12	0

Sbm5

-6	-11	-11	12	12
-11	-18	-18	19	20
-11	-18	17	19	20
-6	-11	10	12	12
0	0	0	0	0

Sbm6

12	-11	-11	-6	0
19	-18	-18	-11	0
19	17	-18	-11	0
12	10	-11	-6	0
0	0	0	0	0

Sbm7

-6	-11	10	12	12
-11	-18	17	19	20
-11	-11	-18	19	20
-6	-11	-11	12	12
0	0	0	0	0

Sbm8

12	10	-11	-6	0
19	17	-18	-11	0
19	-18	-18	-11	0
12	-11	-11	-6	0
0	0	0	0	0

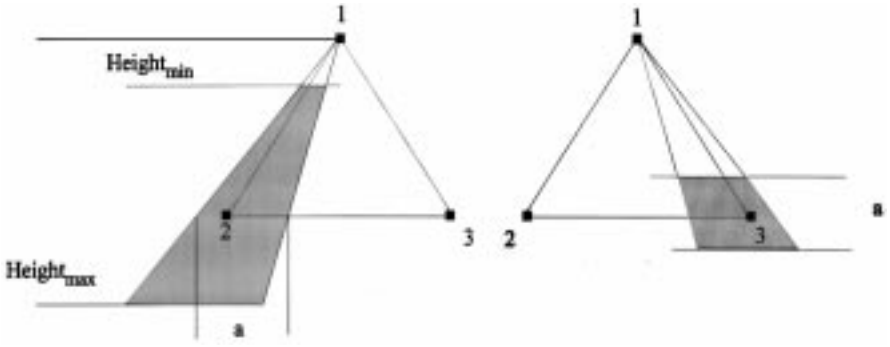

Fig. 4. Triangular sign detection algorithm.

Fig. 5. Types 2 and 3 corner detection.

1) First is corner detection. All the corners that belong to the types that appear in the sign are sought in the image, i.e., upper left vertex (C1 type), upper right vertex (C2 type), lower left vertex (C3 type), and lower right vertex (C4 type).
2) Second is the study of the corner position. The algorithm used for the study of the corner position is similar to the one used for triangular signs (Fig. 6).
   a) The image is scanned until a C1-type corner is found.
   b) From the C1 corner found, a search area is defined through two lines that start from it and have slopes of 85° and 95°, respectively. This area will also be limited by the maximum and the minimum heights that we expect the sign has in the image. In this area, a C2 corner is sought. If it is not found, the algorithm returns to step a), to continue scanning the image, looking for another C1 corner.
   c) A second search zone is created, delimited by two lines that start from the corner of the C2 corner and have slopes of 5° and −5°, respectively, and again, by the maximum and the minimum heights that we expect the sign has in the image. In this second area, a C3-type corner is sought. If it is found, the algorithm considers that the three corners found in the successive stages correspond to the same sign, otherwise it returns to step b).
   d) A third search area is created, delimited by two lines that start from the C3 corner and have slopes of −95° and 85°, respectively, and two lines depending on the position of the C2 corner:

$$a = \text{Height}_{\text{corner2}} - \frac{c}{2} \tag{10}$$

$$b = \text{Height}_{\text{corner2}} + \frac{c}{2}. \tag{11}$$

   In this third zone, a C4 is sought. If it is not found, the algorithm returns to step c).
   e) In this last step, we check if the corners correspond to a rectangle or if, on the contrary, they are points of a circle. The explanation of this step and the reason why it is necessary will be seen in Section III-C-3. The result of applying the algorithm to a real image can be observed in Fig. 8(c).

3) Circular Signs Detection: To detect the circumference that limits the sign, a similar method based on masks must to be used. From the equations obtained when the optimal corner detector was described, we can see that, actually, they can be applied for the detection of other features and not only for corners. Masks to locate some portions of a circumference can be built, and the circumference they belong to can be found from the convolution. However, the number of masks would be very high for several radii of the circumference. However, it is possible to use approximate masks that serve for circumferences of any radio. The masks built for the 90° corners are an approximation of small-circumference arcs located in the 45°, 135°, 225°, and 315° angles. The positive part of the masks remains within the circle, while the negative part coincides with the background. Therefore, the resulting values of the convolution are high (Fig. 7). The main advantage of using these masks is that there is no need for new convolutions to detect the circles, since they have already been made for the detection of the rectangular signs. Since the first steps of the algorithm for rectangular signs cannot differentiate among the points that are, indeed, corners and those which belong to a circle, a new step is in charge of that. To make the difference between a rectangle and a circle in the last stage of the algorithm, three of the four points collected
in the previous steps are taken, and the circumference that passes by them is calculated. If most points belong to the circumference, the detected sign is taken as a circular one;

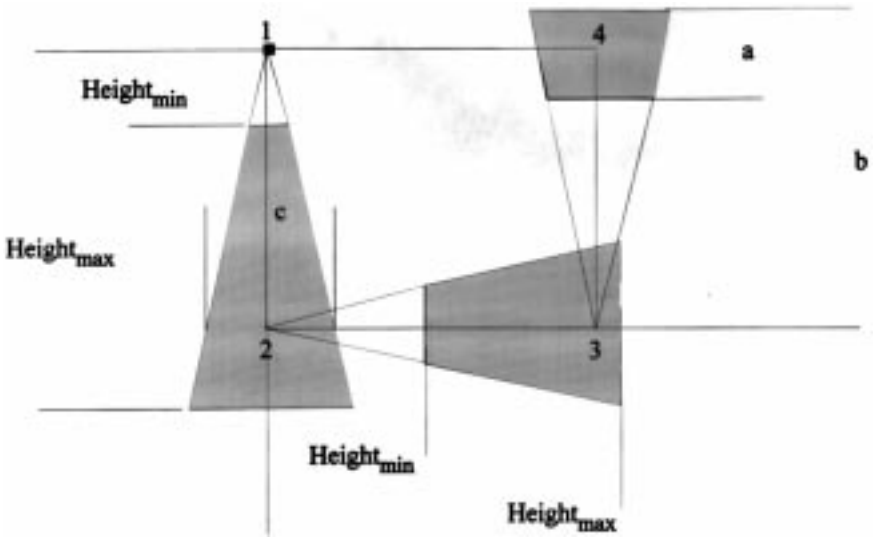

Fig. 6. Rectangular sign detection algorithm.

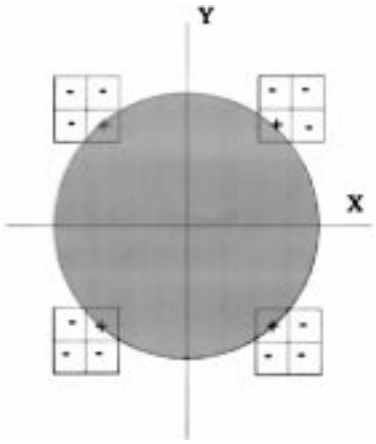

Fig. 7. Circumference detection masks.

otherwise, it is taken as a rectangular one (Fig. 7). The result of applying the algorithm to a real image can be observed in Fig. 8(b).

A consequence of the algorithms described above is that occlusions have not been considered in the detection, and every corner has to be detected. An extrapolation of the method is possible, to suppose from two detected corners where the other one or two should be and to pass to the classification step, but is undergoing research.

TABLE II
IMAGE NORMALIZATION

Image dimension	50			24		
Relation	1.66			0.8		
	Target pixel	x Factor	Original pixel	Target pixel	x Factor	Original pixel
	0	0	0	0	0	0
	1	1.6	2	1	0.8	1
	2	3.2	3	2	1.6	2
	30	50	50	30	24	24

III. TRAFFIC SIGN CLASSIFICATION

Once the presence of a traffic sign is detected, its classification takes place through neural networks. Two options could be taken, either to obtain some features from the inner part of the sign and present them as input patterns or to present the image as the input pattern. The latter was the chosen solution. Two neural networks were trained because the detection algorithm is different according to the form of the sign, i.e., one for circular signs and another for triangular signs. The chosen net was a multilayer perceptron. The size of the input layer corresponds to an image of 30 $\times$ 30 pixels, and the output layer is ten, i.e., nine for the trained signs plus one output that shows that the sign is not one of the nine. The studied nets were three, the number and dimension of their hidden layers being different.

A. Image Normalization

The first step is to normalize the image obtained by the detection module to the dimensions 30 $\times$ 30. To do this, the relation between the dimension we need and the ones we have obtained is calculated, and the pixels are repeated or discarded depending on that relation as shown in Table II.

(a) (b)

(c)

Fig. 8. Real signs detection. (a) Triangular sign detection. (b) Circular sign detection. (c) Rectangular sign detection.

Instead of this nearest neighbor approach, bilinear interpolation has been tried, but with insignificant improvement. As bilinear interpolation is costlier computationally, the nearest neighbor method has been used.

B. Training Patterns

Nine ideal signs were chosen for the net training (Fig. 9). The training patterns are obtained from these signs through the following modifications.

1) We mentioned before that the slope accepted for a sign was $\pm 6°$. From every one of the nine signs, another five were obtained by covering that draft range.
2) Three Gaussian noise levels were added to each of the previous signs. This way, during the training of the net, low weights were associated with the background pixels of the inner part of the sign.
3) Four different thresholds were applied to the resulting image, in order to obtain the information located in the inner part of the sign. In this way, the system is adapted to various lighting conditions that the real images will present.
4) After making a decision about the net dimensions, a new set of training patterns was made, taking into account a displacement of three pixels to the left and to the right.

Then, from the chosen ideal patterns, 1620 training patterns were obtained.

TABLE III
NEURAL NETWORK DIMENSIONS SELECTION

Network	Sign 1	Sign 2	Sign 3	Sign 4
30x30/30/10/10	94	60	76	78
30x30/30/15/10	95	70	96	84
30x30/15/5/10	98	90	98	97

C. Results

The dimensions of the three studied nets are as follows:

1) $30 \times 30/30/10$;
2) $30 \times 30/30/15/10$;
3) $30 \times 30/15/5/10$.

The three networks were trained with the patterns obtained from the first three conditions. In order to compare the results, some test images were chosen, as shown in Fig. 9. The best results corresponded to the third network and are shown in Table III (0 minimum value, 100 maximum).

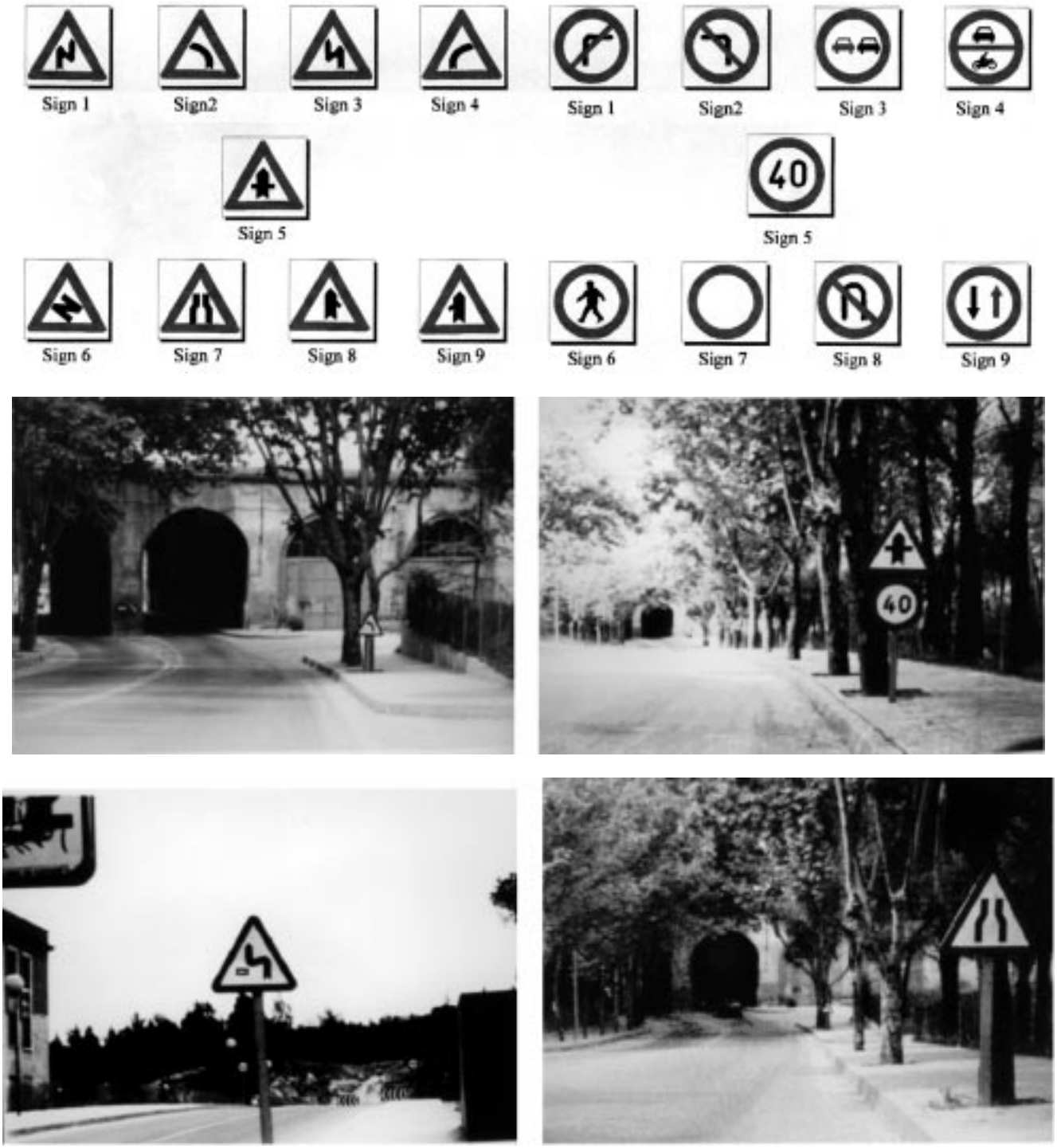

Fig. 9. Ideal signs and test images.

After these results, the dimensions of the last network were chosen to train the network for the circular signs. Additionally, a new training pattern set was created, taking into account the possible displacement of the signal. The output for the images shown in Fig. 10 are shown in Tables IV and V (again, 0 minimum value, 100 maximum).

The algorithm has been implemented in an ITI 150/40 in a PC486 33 MHz with Local Bus. The speed of the detection phase is 220 ms for a 256 × 256 image. The neural network runs in the PC CPU and takes 1.2 s. The implementation of the neural network in a digital signal processor (DSP) is undergoing research, and the expected speed is between 30–40 ms.

TABLE IV
TRIANGULAR SIGNS CLASSIFICATION

	Triangular signs									
	Output 1	Output 2	Output 3	Output 4	Output 5	Output 6	Output 7	Ouput 8	Output 9	Output1 0
Sign1	**98**	1	1	1	1	1	1	1	1	1
Sign2	1	1	1	**96**	1	1	1	1	1	1
Sign3	1	1	**90**	1	1	2	1	1	1	1
Sign4	1	**82**	1	1	8	1	3	1	1	1
Sign5	1	1	1	1	**96**	1	1	1	2	1

Fig. 10. Traffic signs detected and classified.

TABLE V
CIRCULAR SIGNS CLASSIFICATION

Sign6	1	1	1	1	1	**1**	**77**	1	1	1
Circular signs										
	Output 1	Output 2	Output 3	Output 4	Output 5	Output 6	Output 7	Ouput 8	Output 9	Output1 0
Sign7	**96**	1	1	1	1	**1**	1	1	1	1
Sign8	2	1	**97**	1	1	1	2	2	1	1
Sign9	1	1	1	2	**86**	1	2	1	1	1
Sign10	1	1	3	1	**95**	1	1	1	1	1

IV. CONCLUSION

A method for the perception of traffic signs by image analysis has been tested successfully. The algorithm has two main parts, the detection and the classification. For the first part, the color and the corners of the shape of the sign were chosen as features to extract the sign from the environment. It has been proved with different signs and conditions. For the classification, the detected sign was used as the input pattern for a neural network. The multilayer perceptron was chosen. Several networks with different number or layers and nodes were trained and compared. All the algorithms can be achieved in real time with a PC and a pipeline image processing board. Above all, some improvements are the study of partial occlusions and the use of other paradigms of neural networks.

Fig. 10. *(Continued.)* Traffic signs detected and classified.

ACKNOWLEDGMENT

The authors gratefully acknowledge C. Gagnon for her help during the preparation of this paper.

REFERENCES

[1] J. Crissman and C. E. Thorpe "UNSCARF, a color vision system for the detection of unstructured roads," in *Proc. IEEE Int. Conf. Robotics and Automation*, Sacramento, CA, Apr. 1991, pp. 2496–2501.

[2] L. Davis, "Visual navigation at the University of Maryland," in *Proc. Int. Conf. Intelligent Autonomous Systems 2*, Amsterdam, The Netherlands, 1989, pp. 1–19.

[3] E. Dickmans, "Machine perception exploiting high-level spatio-temporal models," presented at the AGARD Lecture Series 185, Madrid, Spain, Sept. 17–18, 1992.

[4] D. Pomerleau, "Neural network based autonomous navigation," in *Vision and Navigation: The Carnegie Mellon Navlab*, C. E. Thorpe, Ed. Norwell, MA: Kluwer, 1990, ch. 5.

[5] I. Masaki, Ed., *Vision Based Vehicle Guidance*. Berlin, Germany: Springer-Verlag, 1992.

[6] R. Luo, H. Potlapalli, and D. Hislop, "Autocorrelation," in *Proc. Int. Conf. Industrial Electronics, Control, Instrumentation and Automation*, San Diego, CA, Nov. 9–13, 1992, pp. 700–705.

[7] ——, "Translation and scale invariant landmark recognition using receptive field neural networks," in *Proc. Int. Conf. Intelligent Robots and Systems (IROS'92)*, 1992, pp. 527–533.

[8] R. Luo and H. Potlapalli, "Landmark recognition using protection learning for mobile robot navigation," in *Proc. Int. Conf. Neural Networks*, Orlando, FL, June 27–29, 1994, pp. 2703–2708.

[9] M. Blancard, "Road sign recognition: A study of vision-based decision making for road environment recognition," in *Vision Based Vehicle Guidance*, I. Masaki, Ed. Berlin, Germany: Springer-Verlag, 1992, pp. 162–175.

[10] G. Piccioli, E. De Micheli, and M. Campani, "A robust method for road sign detection and recognition," in *Proc. 3rd European Conf. Computer Vision*, 1994, pp. 495–500.

[11] B. Bessere, S. Estable, B. Ulmer, and D. Reichardt, "Shape classification for traffic sign recognition," in *Proc. 1st IFAC Int. Workshop Intelligent Autonomous Vehicles*, 1993, pp. 487–492.

[12] S. Estable, J. Schick, F. Stein, R. Jhansen, R. O. H. Writter, and Y.-J. Zheng, "A real-time traffic sign recognition system," in *Proc. Intelligent Vehicles Symp.*, Paris, France, 1994, pp. 212–218.

[13] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 2nd ed. Reading, MA: Addison-Wesley, 1993.

[14] H. Kamada and M. Yoshida, "A visual control system using image processing and fuzzy theory," in *Vision Based Vehicle Guidance*, I. Masaki, Ed. Berlin, Germany: Springer-Verlag, 1992, pp. 111–128.

[15] L. Dreschler and H. Nagel, "Volumetric model and 3-D trajectory of a moving car derived from monocular TV-frame sequence of a street scene," in *Proc. IJCAI*, 1981, pp. 692–697.

[16] M. Shah and R. Jain, "Detecting time-varying corners," *Comput. Vision, Graph., Image Processing*, vol. 28, no. 3, pp. 345–355, Dec. 1984.

[17] L. Kitchen and A. Rosenfeld, "Gray-level corner detection," *Pattern Recognit. Lett.*, vol. 1, pp. 95–102, 1982.

[18] O. A. Zuniga and R. Haralik, "Corner detection using the facet model," in *Proc. IEEE CVPR Conf.*, 1983, pp. 30–37.

[19] K. Rangarajan, M. Shah, and D. Van Brackle, "Optimal corner detector," *Comput. Vision, Graph., Image Processing*, vol. 48, no. 2, pp. 230–245, Nov. 1989.

Arturo de la Escalera (A'96–M'97) was born in Linares, Spain, in 1965. He received the Degree in automation and electronics engineering in 1989 and the Ph.D. degree in industrial engineering in 1995 from the Universidad Politecnica de Madrid, Madrid, Spain.

Since 1993, he has been an Assistant Professor in the Department of Engineering, Universidad Carlos III de Madrid, Madrid, Spain. His research interest is advanced robotics, with special emphasis on vision sensor systems and image data processing methods for environment perception and mobile robot relocalization.

Luis E. Moreno (M'91) received the Degree in electrical engineering in 1984 and the Ph.D. degree in 1988 from the Universidad Politecnica de Madrid, Madrid, Spain.

From 1988 to 1994, he was an Associate Professor at the Universidad Politecnica de Madrid. In 1994, he joined the Department of Engineering, Universidad Carlos III de Madrid, Madrid, Spain, where he has been involved in several mobile robotics projects. He is the author of more than 35 contributions to international conferences, journals, and books. His research interests are in the areas of mobile robotics, sensor fusion, environment modeling, and computer vision.

Miguel Angel Salichs (M'91) received the Degree in electrical engineering 1978 and the Ph.D. degree in 1982 from the Universidad Politecnica de Madrid, Madrid, Spain.

From 1978 to 1991, he was a Member of the Faculty, Universidad Politecnica de Madrid. He is currently a Professor in the Department of Engineering and Head of the Systems Engineering and Automation Division, Universidad Carlos III de Madrid, Madrid, Spain. His research interests include intelligent autonomous systems, mobile robots, perception systems, and intelligent control.

Dr. Salichs currently serves as the Chairman of the Intelligent Autonomous Vehicles Committee of the International Federation of Automatic Control (IFAC).

José María Armingol received the Degree in automation and electronics engineering from the Universidad Politecnica de Madrid, Madrid, Spain, in 1992. He is currently working toward the Ph.D. degree in the Department of Engineering, Universidad Carlos III de Madrid, Madrid, Spain.

He is also currently an Assistant Professor in the Department of Engineering, Universidad Carlos III de Madrid. His research interests are in the areas of image processing and pattern recognition for mobile robot relocalization.