Signal Processing 88 (2008) 2943–2955

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Traffic sign shape classification and localization based on the normalized FFT of the signature of blobs and 2D homographies

Pedro Gil Jiménez *, Saturnino Maldonado Bascón, Hilario Gómez Moreno, Sergio Lafuente Arroyo, Francisco López Ferreras

Dpto. de Teoría de la señal y Comunicaciones, Universidad de Alcalá, Crta. Madrid-Barcelona Km. 33.600, 28805 Alcalá de Henares (Madrid), Spain

ARTICLE INFO

Article history:
Received 12 September 2007
Received in revised form
3 June 2008
Accepted 30 June 2008
Available online 8 July 2008

Keywords:
Traffic sign
Shape classification
Image processing
Blob signature
2D homographies

ABSTRACT

The main goal of a traffic sign recognition system is the detection and recognition of every traffic sign present in the scene. Frequently, the image processing system is divided into three parts, namely, segmentation, detection and recognition. In this work, we will focus on the detection block, dividing it into two sub-blocks that perform shape classification and localization of the sign, respectively. The classification of the shape is performed by means of the signature of the connected components. Object rotations are tackled with the use of the FFT, and the normalization of the object eccentricity improves the performance in the presence of projection distortions. The effect of occlusions are lowered removing the concave parts of the shape. Finally, we propose a novel algorithm, which computes a 2D homography, to re-orientate the sign for further steps, like sign recognition. Experimental results, evaluated using a huge set of randomly generated synthetic images are also given, showing a great robustness of the algorithm to object scaling, rotation, projective deformation, partial occlusions and noise.

1. Introduction

In traffic road scenarios, intelligent video systems have manyfold applications. However, traffic sign detection and recognition systems have been given increasing importance in recent years. Some examples can be found in [1–11]. Applications of these kinds of systems include:

- Autonomous intelligent vehicle: Since the final purpose of traffic signs is traffic regulation, an autonomous intelligent vehicle should include support for traffic sign detection and recognition. A future intelligent vehicle will take advantage of traffic sign recognition systems, along with other functionalities, such as the detection of road lines or obstacles.
- Driving assistance: Although an autonomous intelligent vehicle is far from being a feasible solution at this time, traffic sign recognition systems can help drivers with some basic tasks, such as limiting the speed of the vehicle, or warning the driver that a dangerous situation has been detected.
- Sign maintenance: Until image processing systems became popular, road maintenance, including sign maintenance, has been performed manually for many years, with an operator watching a videotape and writing down the presence and condition of each traffic sign. With this method being a tedious and expensive procedure, road maintenance companies are now replacing human operators with image processing systems, which can be run more efficiently over a computer at lesser cost.

Although many schemes have been proposed in the literature for the purpose of traffic sign detection and recognition, the more common ones comprise a video camera mounted normally at the front of the vehicle, although some systems may include another video

* Corresponding author. Tel.: +34 918856721; fax: +34 918856699.
E-mail address: pedro.gil@uah.es (P. Gil Jiménez).

0165-1684/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.sigpro.2008.06.019

camera at the rear of the vehicle recording the signs from behind, and a video processing system, normally implemented over a PC for maintenance applications, or over specialized hardware in driving assistance applications.

Focusing on the video processing system, the most common approach found in the literature divides the process into three general blocks, normally identified as *segmentation*, *detection* and *recognition*.

- *Segmentation*: The segmentation block generates a number of binary masks to separate the objects of interest from the background. Since the number of colours used in the construction of traffic signs is limited, comprising basically red, blue, black, white and yellow, common implementations use colour information as the main feature to perform the segmentation. After this step, computation of the connected components generates a list of candidate blobs, where some of them are traffic signs, and the rest are supposed to be discarded by the subsequent blocks.
- *Detection*: The next step is the detection of the sign, and this can be done in several ways. A common implementation is the identification of the shape of the blob, and its classification into a small number of reference shapes. Normally, the equilateral triangle, the octagon, the rectangle and the circle are the most common.
- *Recognition*: By merging colour and shape information, the system is now able to determine the meaning of the sign. The last step is the recognition of the sign, performing a search over a database that includes all possible signs. However, the information concerning colour and shape allows the recognition block to focus the search on a smaller range of possible signs.

Section 2 presents previous research in the field of traffic sign recognition, focusing specially on the detection block. Section 3 presents the detection problem and describes a new algorithm that allows the classification of the sign according to its shape. Section 4 describes sign localization, which includes the accurate determination of the position of the sign and the computation of a 2D homography to restore the image of the sign, placing it into a reference position, facing any possible geometric deformation. Section 5 introduces a computer application, which generates random figures to construct several sets of synthetic images that will be used to evaluate the performance of the algorithm described in this paper. The results obtained with these sets are provided in Section 5 and the conclusions about the performance of the algorithm are discussed in Section 6.

2. State of the art

Although a general traffic sign recognition system is composed of several blocks, in this review we will focus solely on works concerning the detection block in traffic sign recognition, specially on those that use shape detection as its main tool.

For instance, [12] focuses in circular sign detection determining the parameters of a circle (center and radius) choosing three different points along the contour of the blob. For the rest of the points, the mean error is computed, and if lower than a given threshold, the object is considered to be a circle. In this method, projection distortions or occlusions are not taken into account.

In [13], an angle-dependent edge filter detects only circular edges while filtering other types of edges. This algorithm deals with projection distortions, but it leaves some problems unresolved like: difficulties in obtaining centered versions of the blob due to problems with occlusions, cropping, contrast, shadows, or the dependency of the edge detection method on the position of the central point.

Genetic algorithms (GA) are frequently used in the task of shape classification. The works [9,14] are some examples of the use of GA search for circular sign detection but none of them are useful with occluded or distorted signs.

In [15], rectangles are detected through horizontal and vertical projections of the edges of the objects and finding the maximum in the histogram of these projections. The nearest sign is detected and recognized but distortions are not treated, although some degree of occlusion is allowed. This method cannot be easily modified for circles or triangles.

The previously presented methods focus only in one shape, but common applications for traffic sign recognition need the identification of every possible sign, regardless of the shape. For this reason, works that allow the detection of any kind of shape will be desirable. In [16], the system is able to detect triangles and circles through the use of predefined templates in different scales. This method does not consider tilted or skewed signs and the images size is limited. In [5], the distance to border (DtB) is defined, allowing the system to determine the shape by means of support vector machines. In [10], templates of different shapes are stored, and each normalized (36×36) blob is compared with rotated versions to make the system invariant to rotations but distortions are not considered and the rotations are limited in number.

In [17], the axial symmetry of the traffic signs is exploited. Thus, a symmetry framework shape distinguishing method is proposed to allow the detection of several kinds of shapes. The idea is to measure proportionality between several horizontal segments taken from candidate blobs from top to bottom. Each shape is characterized using this method and saved in a model database used for comparison. Projection distortion or occlusions are not treated. In [2], an algorithm that analyzes the number, location and kind of corners of the object is used. This allows the identification of triangles, circles and rectangles. Its corners can be identified by the angle they form. The algorithm searches for corners with angles near 60° and 90° present in triangles and rectangles. Circles are treated like rectangles but finding the angles in tangent lines of the shape instead. No deformation or occlusion is treated specifically.

The work described in this paper comes from a previous research implementing the whole sign recognition

system, which can be found in [18]. The main contribution of this work is basically the improvement of the detection block, where the new method developed here has proven to be more successful than the DtBs method, defined in our previous work. Our first approach to shape classification through the signature was developed in [19]. Although essentially the same, improvements basically consist in computation time reduction, the addition of geometric false alarm detection and the novel shape localization method that has brought better results for the recognition block.

3. Shape classification

In our system, the detection process uses the binary masks returned by the segmentation step to determine the shape and locate the position of any possible traffic sign. The shapes considered here are those normally used in the construction of traffic signs, namely, the equilateral triangle, the circle, the rectangle and the octagon. However, since the shape of an octagon can be approximated, with a small error, to the shape of a circle, we will consider the octagon as a circle throughout this paper. Furthermore, since for some kinds of circular signs its content forces the segmentation algorithm to split the sign into two semicircles, the semicircle is also another shape to be considered. As we will deduce later, the shape considered is not exactly a semicircle, but a semiellipse. Fig. 1 gives us an example of the semicircle problem.

The proposed algorithm performs the classification, comparing the signature of each blob returned by the segmentation step, with the signatures of the reference shapes. In this case, we define the signature as the distance from the mass center to the boundary of the object as a function of the angle [20]. In Fig. 2, the reference shapes mentioned above are illustrated along with their corresponding signatures. Although the signature of the object can be computed directly from the blob itself, some considerations should be taken into account to face unavoidable geometric distortions that can worsen the shape classification success probability. The geometric distortions considered in this paper are:

- *Projection distortions*: Camera projection when the sign is not parallel to the image plane can lead to geometric projection distortions. Under such circumstances, for instance, a circle is imaged as an ellipse, an equilateral triangle as an scalene one, or a rectangle as a general quadrilateral.
- *Scaling*: The size of the projection of the sign in the image plane depends on the distance from the camera to the sign. As the camera approaches the sign, the blob corresponding to the object get bigger, and so, the computation of the signature for the same sign at different distances would yield signatures with different amplitudes.
- *Rotations*: Differences in tilt angles between the image plane and the sign are reflected as object rotations. These rotations become circular shifts in the signature.
- *Occlusions*: Partial occlusions of the sign alters the shape of the blob. This is more likely to happen within cities than on non-urban roads.
- *Camera noise*: Camera noise can be seen as a segmentation problem in such a way that the mask returned by the segmentation block is not a shape with perfect edges, but a version with its edges altered by noise.

In order to improve the shape classification performance, the proposed algorithm does not compute the signature directly from the blob itself, but a modified version which tries to overcome, as far as possible, the geometric distortions considered above. In the following subsections, it is described the solution adopted to deal with each one of the previous problems.

3.1. Projection distortions

A projective distortion happens whenever a plane, for instance, the traffic sign, is imaged by a general camera. Although scaling and rotations can also be considered as projection distortions, these ones are more specific transformations, and will be analyzed in subsequent subsections. Although at this point it is not possible, nor absolutely necessary, to undo every distortion undergone by the sign, it is, however, desirable to perform a geometric transformation of the blob in order to improve the shape classification process.

The main drawback of a projective transformation for our purpose of shape classification from the signature of the blob is the deformation of the object, or more formally, the variation of the aspect ratio, or the

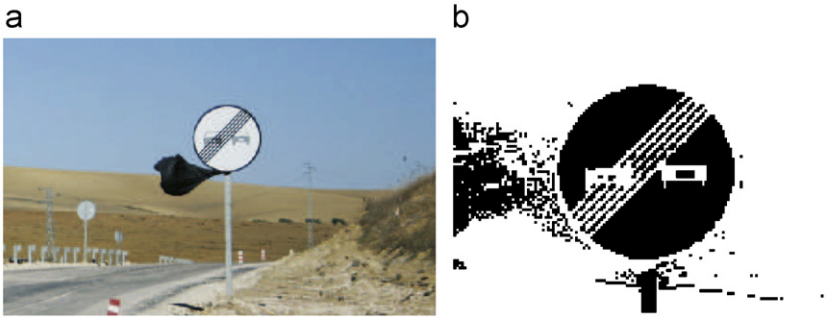

Fig. 1. Semicircle example.

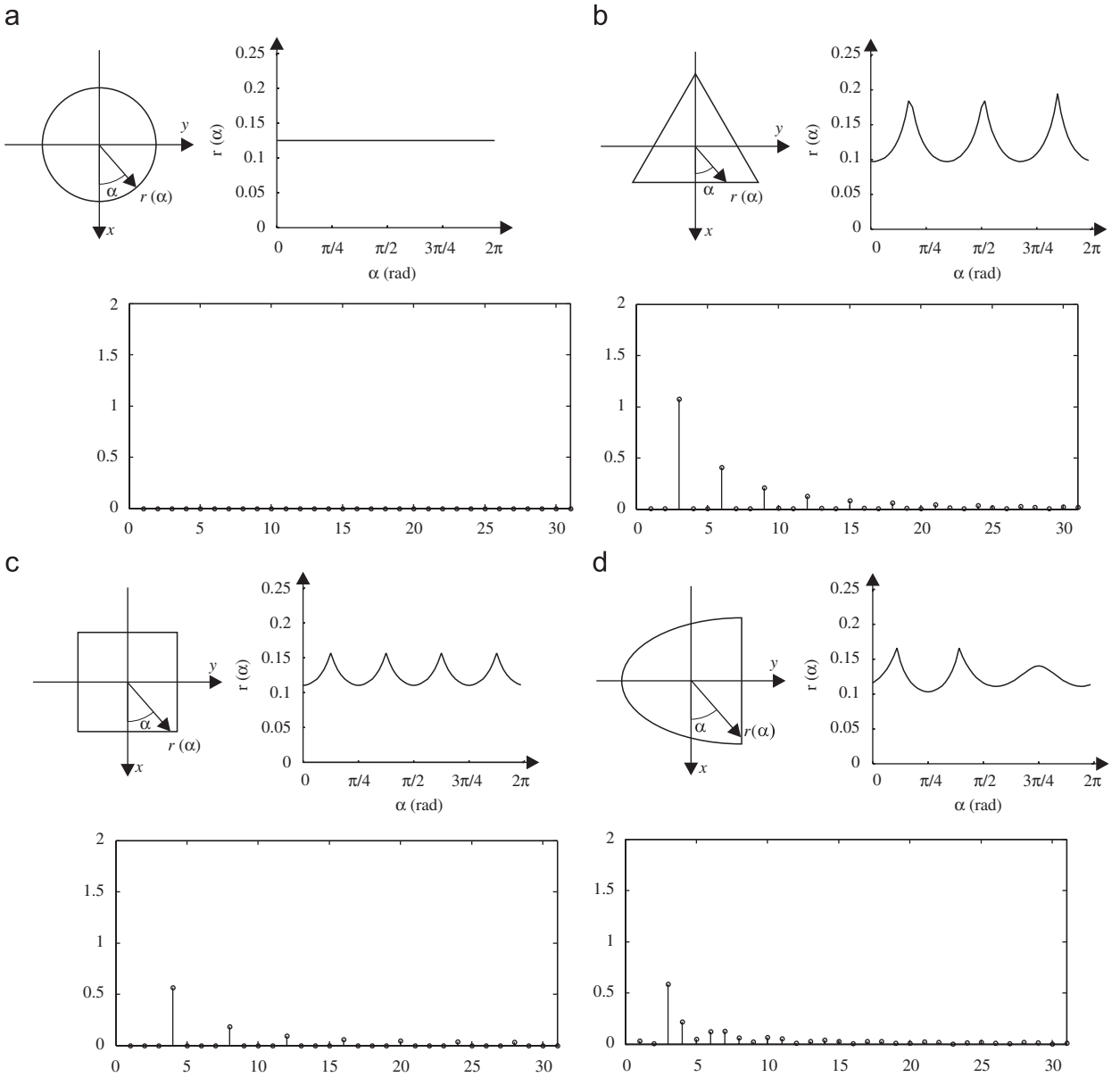

Fig. 2. Reference shapes. (a) Circle, (b) Triangle, (c) Square, (d) Semiellipse.

eccentricity, of the object, than will lead, quite probably, to a classification error. In order to improve the performance of the classifier, a normalization of the aspect ratio of the object is performed for each blob analyzed. To normalize it, first of all, we compute the orientation of the object according to [21]

$$\theta = \frac{1}{2}\arctan\left(\frac{2\cdot\mu_{11}}{\mu_{20}-\mu_{02}}\right), \tag{1}$$

where μ_{11}, μ_{20} and μ_{02} are the central moments of the object. Once the orientation is obtained, a new coordinates system is created, with u being the coordinate in the orientation direction, and v the coordinate orthogonal to coordinate u, as shown in Fig. 3(a). From now on, we denote μ^{uv} as a moment in the uv space, and so on. The transformation matrix in this case is

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}. \tag{2}$$

In this new coordinates system (Fig. 3(b)), the second-order moments μ^{uv}_{20} and μ^{uv}_{02} can be computed from the moments in the xy coordinates system according to these expressions:

$$\mu^{uv}_{20} = m^{uv}_{20} - m_{00}\cdot(\mu^{uv}_{10})^2, \tag{3}$$

$$\mu^{uv}_{02} = m^{uv}_{02} - m_{00}\cdot(\mu^{uv}_{01})^2, \tag{4}$$

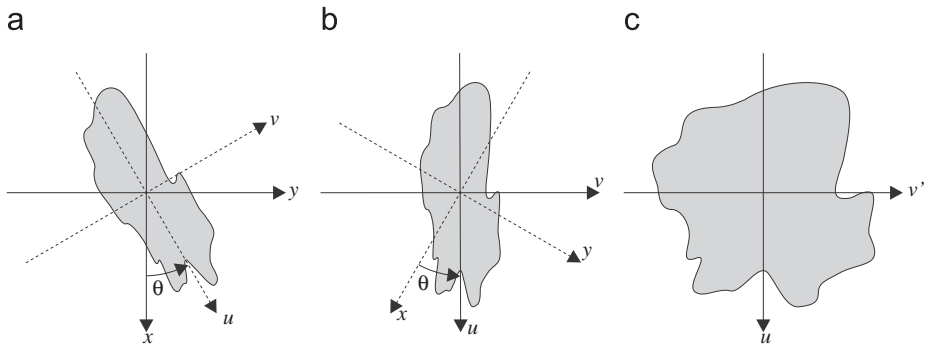

Fig. 3. Coordinates systems.

where

$$m_{20}^{uv} = m_{20}^{xy} \cdot \cos^2\theta + m_{02}^{xy} \cdot \sin^2\theta + 2 \cdot m_{11}^{xy} \cdot \cos\theta \cdot \sin\theta, \quad (5)$$

$$m_{02}^{uv} = m_{20}^{xy} \cdot \sin^2\theta + m_{02}^{xy} \cdot \cos^2\theta - 2 \cdot m_{11}^{xy} \cdot \cos\theta \cdot \sin\theta \quad (6)$$

and

$$\mu_{10}^{uv} = \mu_{10}^{xy} \cdot \cos\theta + \mu_{01}^{xy} \cdot \sin\theta, \quad (7)$$

$$\mu_{01}^{uv} = -\mu_{10}^{xy} \cdot \sin\theta + \mu_{01}^{xy} \cdot \cos\theta. \quad (8)$$

We can define the eccentricity of the object as the relationship between the above computed moments μ_{20}^{uv} and μ_{02}^{uv} as

$$k = \sqrt{\frac{\mu_{02}^{uv}}{\mu_{20}^{uv}}}. \quad (9)$$

This expression will always give a value of k smaller than 1, since this is imposed by the orientation angle examined before. The normalization can be achieved *stretching* the object in the v-axis, as presented in Fig. 3(c). This stretching can be performed using the following matrix transformation:

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -k\sin\theta & k\cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}. \quad (10)$$

The computed signature now belongs to an object with an eccentricity, as defined above, equal to 1 and therefore, the same value needs to be imposed on the reference shapes. It can easily be shown that this relationship is 1 for the first three reference shapes, that is, the ratio between the second-order moment in the x direction and the second-order moment in the y direction, as shown in Fig. 2, is always 1 for a circle, a square and an equilateral triangle. However, this is not true for a semicircle, so the shape needs to be scaled to construct a semiellipse where the relationship between its second-order moments is equal to 1. This semiellipse is shown in Fig. 2(d).

It is important to note here that this correction is intended only to improve the classification of the current object using the signature of the reference objects. The reorientation and normalization of the sign is not a function of this part of the detection step, since this is performed after the shape of the sign has been determined, and it is different for the triangle, square, circle and semicircle. This process is fully described in Section 4.

3.2. Object scaling

Object scaling is achieved through amplification or attenuation of the signature. To make the algorithm invariant to scaling, energy normalization must be performed before we can compare the current signature with the ones we use as references. Obviously, the signatures of the reference shapes previously computed and saved have also need to be energy normalized.

3.3. Object rotations

Object rotations become circular shifts in the object signature. To allow the system to correctly classify the shape under object rotations, instead of comparing directly the signature itself, the comparison is performed over the absolute value of the discrete Fourier transform (DFT), taking advantage of its invariance to shifts. Therefore, the absolute values of the DFT of the normalized signature for the reference shapes are previously computed and stored, and the absolute value of the DFT is computed for each object for further classification.

3.4. Occlusions

Occlusions is one of the biggest challenges in traffic sign recognition system, specially on urban environments, where objects like poles, buildings, cars, or even other signs, can occlude a traffic sign. If the occlusion is quite significant, for instance, the sign is split into two halves, the sign is likely to be missed, unless a special algorithm is designed to deal with this kind of problems. When the occlusion is not big enough, however, some modifications on the original approach can improve the performance of the system. In the case of slight occlusions, we can distinguish between two different kinds of them, specially for the cases of triangles and rectangles: those in which one corner of the shape is removed (see Fig. 4(a)), and those in which the occlusion affects only at the straight part (see Fig. 4(b)).

While the first kind of occlusions are more difficult to deal with, the second are very easy to overcome. In our

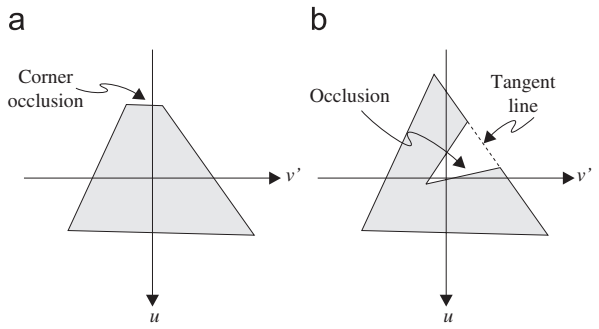

Fig. 4. Occlusion examples.

work, we take advantage of the fact that the shapes considered are all convex, and so, we can cancel a number of occlusions by eliminating all concave parts of the shape tracing tangent lines to the contour of the object at these points. For instance, in Fig. 4(b) we can see how the occlusion is removed, and the result is the same as if there had been no occlusion.

Although this process does not imply any underlying error when the shapes considered are all convex, as it is the case for triangles, rectangles, ellipses and semiellipses, other systems designed for the detection of concave shapes, like crosses or stars, would not take advantage of this process, and a different approach should be designed.

For the cases where one corner (or more) of the shape has disappeared as a consequence of an occlusion, the previous scheme does not solve the problem, since the resulting shape is still convex. In this case, as long as the occlusion is not big enough, the comparison with the reference shapes will yield the proper classification. If the occlusion is quite large, the results are unpredictable.

For the circular case, depending on the degree of occlusion, the shape can become a semicircle, causing a misclassification. Being that an error in this step, for the task of traffic sign recognition this is not a problem, since, as we have seen before, a semicircle always derives from a circle, and so, whenever a semicircle has been detected, the next step is the computation of its generating circle, as we will see later.

3.5. FFT analysis

According to all the details explained above, the actual vector entering the shape classifier is not the signature itself, but the absolute value of the DFT of the normalized signature of an object, derived from the original one. Computation time can be reduced if we fit the number of samples to a power of 2, so that the FFT may be computed instead of the DFT. In this work, the number of samples chosen was 64 (2^6), which is a tradeoff between precision and computational cost. In Fig. 2 it is shown, for each reference shape, the absolute value of the first half of its corresponding 64-samples FFT. In order to make these figures clearer, and taking into account that the sample in 0 is much higher than the rest of the samples, and quite similar for all the shapes (around $\sqrt{64}=8$ for the signatures considered), we have removed this sample from these figures.

Although many kind of classifiers could be used here to perform the shape classification at this point, a simple nearest neighbor (NN) algorithm using the square Euclidean distance as the error measure proves to be enough for our purpose. Furthermore, as we can see in Fig. 2, only about one-quarter of the samples of the FFT are significant, while the rest of the samples are very low, and their value for a real object will be more affected by noise than by the actual shape. The same can be said about the sample in 0, which is quite similar for the four shapes and then, it does not provide any information. So, the final algorithm performs the classification computing the square Euclidean distance only for samples 1–8 of the 64-sampled FFT.

4. Shape localization

Normally, a sign has to be located in a reference position for proper recognition subsequent to detection. This means the system has to face any possible rotation, translation, scaling, or projective deformation before the object enters the next block. Therefore, normalization for every object must be performed to place the object in a proper reference position. In our system, we can take advantage of the fact that the shape of the object is already known so that a normalization algorithm can be designed independently for each shape. We will describe the algorithms for the normalization of the different shapes later in this section.

Independent of the shape of the object, the image of the sign can be considered as a 2D homography, mapping points from the plane containing the sign to another plane, i.e., the image plane [22]. Given this fact, the normalization problem is reduced to the computation of a 2D homography, that is, a 3×3 non-singular matrix H. However, this computation is highly dependent of the shape. For instance, for triangles and rectangles, this matrix can be computed estimating the coordinates of the vertices of the shape, and then establishing correspondences between these points and the position of the vertices of a reference shape. For circles and semicircles, this computation must be done estimating the parameters of the corresponding ellipse. For that reason, since the computation of the homography is quite different for each shape, in the following subsections, the specific algorithm for each shape is developed.

4.1. Triangle normalization and reorientation

A 2D homography can be represented by a 3×3 non-singular matrix with eight degrees of freedom (DOF), accounting for the nine elements of a 3×3 non-singular matrix less one for scale. A common way of computing this homography is through the direct linear

transformation (DLT) algorithm [22]. The DLT algorithm needs a set of point correspondences, and the minimum number of such correspondences depends on the kind of homography being performed.

In the 2D general case, eight DOF have to be computed. Since each point-to-point correspondence accounts for two constraints, a minimum of four point correspondences is needed. However, some assumptions can be imposed in the imaging process, reducing the number of DOF in the system. Although the general case of a camera imaging a plane is a projectivity, which accounts for the eight DOF mentioned above, a typical recording system employed in traffic sign recognition can be approximated to an affinity. Thus, by reducing the number of DOF to only six, the number of point correspondences is lowered to three. In theory, affine simplification occurs when a plane is imaged by a camera that is at infinity. Therefore, affine transformation can be supposed whenever the camera is at infinity, or at least at a great distance from the plane with respect to the size of the sign, which is normally the case for the traffic sign recognition problem discussed here. The matrix $\mathtt{H}$ then becomes

$$\mathtt{H}_A = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix}. \tag{11}$$

For the triangle, these point correspondences can be extracted directly from the three corners of the triangle, and so, the problem is reduced to the identification of these three points in the original image. As we can see in Fig. 2(b), the corners of the triangle could be found using the previously computed signature, since the corners correspond to the peaks of the signature.

Although this method would be a good strategy in the theoretical case, when applied to real images, the results become very inaccurate because we rely on the information of only one pixel, which can be corrupted with noise. Furthermore, if an occlusion makes one of the vertices disappear, the estimated coordinates of the hidden vertex will be located at a very different position. We also need to consider that the system does not yield sub-pixel precision, since the output of that algorithm would be the position of the 3 pixels corresponding to the peaks of the signature.

These problems may be overcome if we consider more than one pixel. Therefore, instead of determining directly the coordinates of the pixels corresponding to the vertices, we can estimate the parameters of the three straight lines that compose the triangle. This estimation can be performed using a number of pixels from the contour of the blob. The easiest way of doing this is using directly the samples of the signature. Considering that the three peaks of the signature have already been localized, all the pixels between two peaks compose one straight line. Using homogeneous coordinates, a point $\mathbf{x_i} = (x_i, y_i, 1)^{\mathrm{T}}$ belongs to a straight line $\mathbf{l} = (l_a, l_b, l_c)^{\mathrm{T}}$ if

$$(x_i, y_i, 1)(l_a, l_b, l_c)^{\mathrm{T}} = 0. \tag{12}$$

Stacking the equation for all the available points for a particular line, we get

$$\mathtt{X} \cdot \mathbf{l} = \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_N & y_N & 1 \end{pmatrix} (l_a \;\; l_b \;\; l_c)^{\mathrm{T}} = \mathbf{0}, \tag{13}$$

which implies the computation of the null-space of matrix $\mathtt{X}$. Since the number of rows of $\mathtt{X}$ is, in general, greater than two, and are affected by noise, (13) defines an overdetermined system.

This system must be solved using some minimization techniques. Although iterative geometric-distance minimization techniques should be used to get the most accurate solution, simple algebraic-distance minimization is accurate enough for our purpose, as we can take advantage of its lesser computational complexity. Geometric-distance minimization is suitable in the presence of outliers, or when the data are highly affected by noise. Note, however, that in our case, neither are there outliers nor is the noise high. Otherwise, the shape detection algorithm would have failed and none of this would make sense.

At this point, we can estimate the degree of confidence of the computed line. Computing the mean geometric error as the mean of the sum of the distances from each point of the signature to the estimated line according to

$$e = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{l_a x_i + l_b y_i + l_c}{\sqrt{l_a^2 + l_b^2}} \right|, \tag{14}$$

allows us to evaluate to what degree the list of points correspond to the line. The threshold can be set for the mean geometric error of each line, and if one of the errors exceeds the given threshold, the blob can be considered as a false alarm. That is, the shape classification algorithm classified the blob as a triangle but, actually, the blob is not a triangle because it is not composed of straight lines, according to our threshold.

Once the three lines of the triangle have been correctly obtained, the vertices of the triangle can be

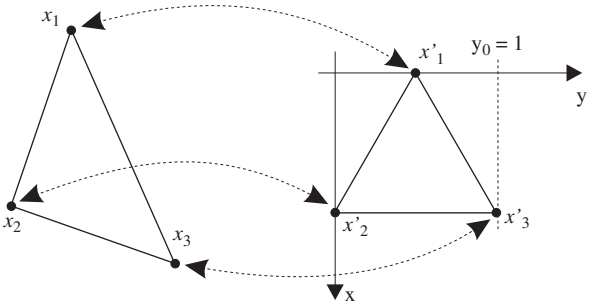

Fig. 5. Triangle correspondences.

computed using

$$\mathbf{x}_{ij} = \mathbf{l}_i \times \mathbf{l}_j = \begin{vmatrix} a & b & c \\ l_{ia} & l_{ib} & l_{ic} \\ l_{ja} & l_{jb} & l_{jc} \end{vmatrix} = \begin{pmatrix} l_{ib}l_{jc} - l_{ic}l_{jb} \\ l_{ic}l_{ja} - l_{ia}l_{jc} \\ l_{ia}l_{jb} - l_{ib}l_{ja} \end{pmatrix} \tag{15}$$

for $(i,j) = \{(1,2),(2,3),(3,1)\}$, and so, the three vertices have been estimated. Once the three corners have been located, we can make the points correspondence to the reference shape, as shown in Fig. 5, where the reference shape is an equilateral triangle of size 1. One assumption has to be made at this point. As we can see in Fig. 5, each corner of the original triangle has to be identified in the reference triangle, and if no other constraints existed, we would have three possibilities. Over normal operation conditions, we can assume that any shape never undergoes a rotation greater than 60°, and so, the top vertex of the original triangle would always correspond to the top vertex of the reference triangle. However, if in a specific application this assumption cannot be imposed, the problem is easily solved in the recognition step by simply generating three different normalized images: one of them being the image yielded by the homography computed in this section, and the other two being rotated versions of the former, one at +120°, and the other at −120°. With these assumptions, three point correspondences $\mathbf{x_i} \leftrightarrow \mathbf{x}'_i$ $(i = 1,2,3)$ are obtained. Writing $\mathbf{x_i} = (x_i, y_i, 1)^{\mathrm{T}}$ and $\mathbf{x}'_\mathbf{i} = (x'_i, y'_i, 1)^{\mathrm{T}}$, the DLT algorithm for an affine transformation gives us the following linear system for each point-to-point correspondence [22]:

$$\begin{bmatrix} 0 & 0 & 0 & -x' & -y' & -1 \\ x' & y' & 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{pmatrix} \mathbf{h^1} \\ \mathbf{h^2} \end{pmatrix} = \begin{pmatrix} -y \\ x \end{pmatrix}, \tag{16}$$

where $\mathbf{h^1} = (h_{11}, h_{12}, h_{13})^{\mathrm{T}}$ and $\mathbf{h^2} = (h_{21}, h_{22}, h_{23})^{\mathrm{T}}$ are the first two rows of the affine matrix H_A forming a 6-vector. The last equation can be written compactly as

$$\mathrm{X}\mathbf{h} = \mathbf{y}. \tag{17}$$

Stacking the six equations obtained from the three correspondences, we get a linear system of six unknowns and six equations, which can be solved for $\mathbf{h}$ using standard techniques for solving linear equations.

4.2. Rectangle normalization and reorientation

Few modifications need to be made to the preceding algorithm to compute the reorientation matrix for a rectangle. In the case of a rectangle, four points are available to compute the projection matrix. Although these four point-to-point correspondences would allow us to compute the projective homography with its eight DOF, the affine transformation assumption is also valid here with little error, allowing the system to compute the homography faster. Thus, we have an over-determined linear system that can be solved by minimizing the mean square error, using the normal equations:

$$(\mathrm{X}^{\mathrm{T}}\mathrm{X})\mathbf{h} = \mathrm{X}^{\mathrm{T}}\mathbf{y}. \tag{18}$$

The four corners of the rectangle are found in the same way as in a triangle, using the signature previously computed, and the four correspondences are established as shown in Fig. 6. Again, the same problem appears here, in the sense that, for a given point, four different correspondences may be chosen, and again, a similar supposition is discussed here. In the case of a rectangle, we can assume that, over normal operation conditions, the shape will not undergo a rotation greater than 45°. Therefore, we can assert that the bottom right corner of the reference rectangle corresponds to the first peak in the signature, that is, the first corner we would find if we had started from the *x*-axis, advancing counter-clockwise. The same solution as in the triangle case can be used here when a specific application does not ensure that the objects will not undergo a rotation greater than 45°. In this case, four different normalized images must be obtained, the other three being the rotated versions computed at 90°, 180°, and 270°.

4.3. Circle normalization

A considerable drawback appears in the computation of the projection matrix for a circle, since a circle does not have corners, and thus, no reference points are available. For this reason, there are no point correspondences, and the homography must be computed in a different way. Taking into account that an affine transformation can map a circle into an ellipse, we would only need to estimate the parameters of the ellipse currently analyzed to perform the computation of the corresponding homography. It is important to stress that an ellipse can be defined with five parameters, that is, its center, which accounts for two DOF, the minor and major radius and its orientation, while the affine homography has six DOF. The last DOF corresponds to the circle orientation, which cannot be computed due to the lack of reference points, and so, the homography can be estimated only up to a circle rotation around its center.

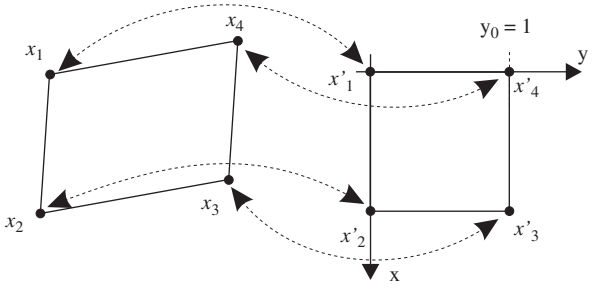

Fig. 6. Rectangle correspondences.

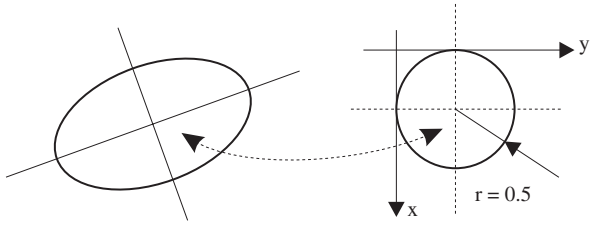

Fig. 7. Circle correspondences.

Although many algorithms have been described in the literature for fitting points to ellipses [23], they focus on the computation of the parameters of the ellipse. Our approach, however, computes directly the homography required for the re-orientation of the shape without any other computation. This transformation maps the original ellipse into the reference shape, in this case a circle of diameter 1, and its center (0.5, 0.5), as shown in Fig. 7.

Taking into account only blob contour pixels, or equivalently, the samples of the signature, we can consider the ellipse as a conic [22] that can be described by a symmetric matrix of the form

$$\mathrm{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}, \tag{19}$$

with five DOF accounting for the six elements of a symmetric matrix less one for scaling. Therefore, a minimum of five points is required to define an ellipse. Any point lying on the ellipse must fulfill:

$$\mathbf{x}_i^{\mathrm{T}} \mathrm{C} \mathbf{x}_i = 0, \tag{20}$$

or equivalently:

$$(x_i^2, x_i y_i, y_i^2, x_i, y_i, 1)\mathbf{c} = 0, \tag{21}$$

where $\mathbf{c} = (a, b, c, d, e, f)^{\mathrm{T}}$ is the conic C represented as a 6-vector. Stacking the equation for all the available points of the signature of the blob, we get the following system:

$$\begin{pmatrix} x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & x_2 y_2 & y_2^2 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_N^2 & x_N y_N & y_N^2 & x_N & y_N & 1 \end{pmatrix} \mathbf{c} = \mathbf{0}. \tag{22}$$

If the number of points N is greater than 5, (22) implies the computation of the null-space of an over-determined system, which can be solved easily using algebraic-distance minimization techniques. Once the conic C has been computed, the following step is the computation of a 2D homography, which transforms the ellipse into an auxiliary circle of radius 0.5 centered at the coordinates' origin, according to [22]

$$\mathrm{C}' = \mathrm{H}^{-\mathrm{T}} \mathrm{C} \mathrm{H}^{-1}, \tag{23}$$

where $\mathrm{C}' = \mathrm{diag}(4, 4, -1)$ is the auxiliary circle represented as a 3×3 diagonal matrix with diagonal entries 4, 4 and -1. Taking the inverse transformation, and decomposing H into a product of matrices:

$$\mathrm{C} = \mathrm{H}_{\mathrm{R}}^{\mathrm{T}} \mathrm{H}_{\mathrm{P}}^{\mathrm{T}} \mathrm{H}_{\mathrm{D}} \mathrm{C}' \mathrm{H}_{\mathrm{D}} \mathrm{H}_{\mathrm{P}} \mathrm{H}_{\mathrm{R}}, \tag{24}$$

where H_{R} is an orthogonal matrix. Furthermore, $\mathrm{C} = \mathrm{H}_{\mathrm{R}}^{\mathrm{T}} \mathrm{C}_{\mathrm{D}} \mathrm{H}_{\mathrm{R}}$, where C_{D} is a diagonal matrix, can be seen as the singular value decomposition (SVD) of C, which can be solved using standard SVD algorithms. Matrix H_{P} is a permutation matrix to ensure that $\mathrm{C}_{\mathrm{D}} = \mathrm{H}_{\mathrm{P}}^{\mathrm{T}} \mathrm{C}_{\mathrm{E}} \mathrm{H}_{\mathrm{P}}$, being C_{E} a diagonal matrix with entries e_{11} and e_{22} positive, and entry e_{33} negative. Finally, H_{D} is a diagonal matrix with entries:

$$d_{11} = \sqrt{e_{11}}/2, \quad d_{22} = \sqrt{e_{22}}/2, \quad d_{33} = \sqrt{-e_{33}}. \tag{25}$$

Matrix $\mathrm{H} = \mathrm{H}_{\mathrm{D}} \mathrm{H}_{\mathrm{P}} \mathrm{H}_{\mathrm{R}}$ is then the 2D homography matrix that transforms the original ellipse C into the circle C' of radius 0.5 centered at the origin of coordinates. We can measure the mean geometric error as the mean of the sum of the distances from each point of the signature to the estimated ellipse according to

$$e = \frac{1}{N} \sum_{i=1}^{N} |d(\mathrm{H}\mathbf{p}_i, \mathbf{0}) - 0.5|, \tag{26}$$

where $d(\mathbf{x}_1, \mathbf{x}_2)$ is the Euclidean distance between points $\mathbf{x}_1$ and $\mathbf{x}_2$, $\mathbf{p}_i' = \mathrm{H}\mathbf{p}_i$ are the original points of the signature mapped to the reference plane, $\mathbf{0}$ are the coordinates of the circle center $(0, 0)$, and 0.5 is the radius of the circle. The mean geometric error, then, can be set at a threshold to discard false alarms. Finally, since the reference figure is a circle of radius 0.5 centered at $(0.5, 0.5)$, we need a translation matrix to shift the previous circle into the reference one:

$$\mathrm{H}_t = \begin{pmatrix} \mathrm{I} & \mathbf{t} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix}, \tag{27}$$

where I is a 2×2 identity matrix and $\mathbf{t} = (0.5, 0.5)^{\mathrm{T}}$ is the translation vector.

4.4. Semicircle normalization

If we consider the semicircle case as a segmentation problem in which, the segmentation step returns two semicircles instead of a whole circle, the easiest way to overcome this problem in the detection step is by estimating the circle from which the semicircles originated. This process has another advantage, namely, if due to some problems in the segmentation or detection step, only one semicircle was detected, while the other was lost, the valid semicircle is still enough to estimate the whole circle, and therefore, recognize the sign. If both semicircles were detected, the traffic sign could be detected twice, increasing the robustness of the system. Therefore, the semicircle problem is reduced to the computation of the homography that transforms the complete circle into a reference one, exactly the same problem described in the previous section.

Although the points to estimate the homography can be extracted from the signature, the problem now lies in the decision about which points of the signature belong to the circle part of the semicircle, and which points are part

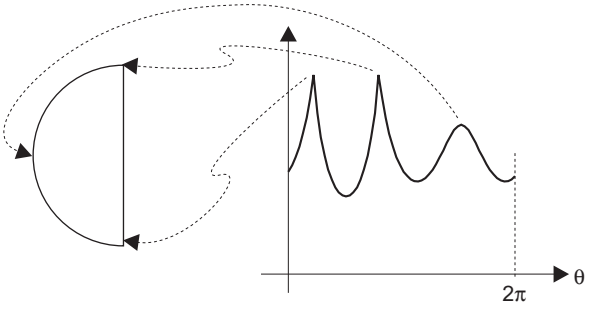

Fig. 8. Semicircle correspondences.

of the straight line. A glance at Fig. 8 shows that the signature of a semiellipse has three peaks, two of which correspond to the two cross points of the ellipse with the straight line, and the other one that corresponds approximately to the central point between the last two points.

Based on this discussion, we can execute an algorithm similar to the one designed for the triangular case, and use it to estimate the three lines joining the points that correspond to the peaks of the signature. If we compute the geometric error for each line according to (14), we should find one of them with the smallest error, whereas the other two must have greater errors. Once we have determined which part of the signature corresponds to the straight line of the semicircle, and which corresponds to the circular part, we simply have to construct a linear system, similar to the one described in (22), using only the samples of the signature corresponding to the circular part. The rest of the algorithm is then similar to the one described in the previous section.

Two evaluations can be done to estimate the degree to which the blob corresponds to a semicircle. In the first step, once the geometrical error for the three lines have been computed, and the line which corresponds to the straight part of the semicircle is determined, we can set a threshold of geometric error for this line, in the same way we did for triangles and rectangles. If this error exceeds the given threshold, we can conclude that this line is not a straight line, and so, the blob is marked as a false alarm. In the second step, as we did for the circular case, we can compute the degree to which the points corresponding to the circular part of the signature fit the estimated circle, and again, set a threshold of this error in order to detect false alarms.

5. Results

So far, the main characteristics concerning the traffic sign detection problem have been described, and a new algorithm has been proposed. The algorithm was implemented in C using Visual C++ 6.0. This implementation also includes a segmentation and a recognition block, to allow the execution of the complete recognition system on real images. A complete description of these blocks can be found in [1].

Although the complete system can work with real images, we have tested the algorithm using synthetic images with figures generated automatically in order to make the results independent of the segmentation block, and even of camera quality or illumination conditions. This process also has the advantage of being able to store the shape and exact position of a figure at the same time that it is generated. This information can be used to automatically evaluate the performance of the proposed algorithm. Therefore, a computer application that can automatically generate triangles, rectangles, ellipses and semiellipses has been implemented. For the construction of triangles, the tool generates three random coordinates, and, after validating some geometric constraints, a triangle, having as vertices the three random coordinates, is drawn. For the rectangles, the tool works quite similarly.

Fig. 9. Successful examples of noisy synthetic figures with σ equal to: (a) 6 pixels; (b) 9 pixels.

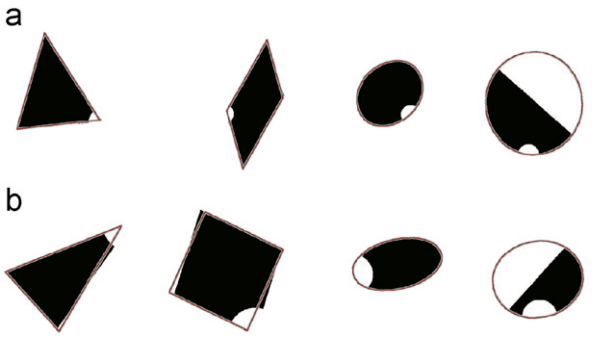

Fig. 10. Examples of synthetic figures with occlusions, with percentage of occlusion equal to: (a) 15%; (b) 30%.

However, we have to note that, although any kind of quadrilaterals can be constructed, only simple convex ones would make sense for the traffic sign detection problem. Furthermore, since only affine transformations have been defined, only parallelograms must be considered. For ellipses, five random numbers must be generated, two for the center, one for the major axis, one for the minor axis and the last for the orientation of the major axis. For semiellipses, one additional random number is generated, the one that defines the orientation of the straight line which divides the ellipse into two halves.

We also want to determine the behavior of the algorithm in the presence of noise. To simulate this noise, a second step has been added to the application described above. Basically, this step generates several random coordinates near the edges of the synthetic figures, and draws circular patches of random diameters, which alter the contour of the shape. To allow the evaluation of the algorithm in the presence of different levels of noise, the diameter of the patches is a random variable with typical deviation σ, configurable in such a way that we can generate several sets of figures with different values of σ each. Obviously, the greater the diameter of the patches, the more distorted the final figure will be compared to the original one. Fig. 9 shows some examples of noisy figures generated with this application.

The effect of partial shape occlusion is another problem to be analyzed here. To this end, a bigger circular patch of a configurable diameter can also be drawn on the edge of the figure to simulate this effect. For triangles and

rectangles, this patch is drawn just on one of the vertex of the shape chosen at random, since that is worse than a similar patch drawn on the straight part of the edge of the shape. For circular shapes, the patch is drawn around one random point on the circumference. The same can be applied to semicircles, except that the application needs to make sure that point does not belong to the deleted part of the circle. Some examples of occluded figures are shown in Fig. 10. The complete set of images used in this section for the evaluation of the proposed algorithm can be found in http://agamenon.tsc.uah.es/Investigacion/gram/papers/Elsevier07.

With this additional tool, we can now evaluate the performance of the proposed algorithm. Two different parameters are evaluated in this step. On the one hand, the classification success probability is measured as the ratio between the number of correctly classified shapes and the total number of figures analyzed. On the other hand, the accuracy in the localization of the position of the shape is also measured. To this end, the original image, before the noise or occlusion was added, is compared with the estimated figure, and the non-coincident pixels, i.e., pixels that in the original image belong to the figure while in the estimated image do not or vice versa, are counted. We define the percentage of area error as the ratio between the sum of all non-coincident pixels for all the figures of the same shape and level of noise or occlusion, and the sum of the pixels of all the original figures. Obviously, this ratio is only applicable to correctly classified shapes, without taking into account misclassified ones.

Table 1
Numerical results

	$\sigma = 0$		$\sigma = 5$		$\sigma = 10$	
	% Succ.	% A. Err.	% Succ.	% A. Err.	% Succ.	% A. Err.
Triangle	100	1.20	96.80	9.40	53.80	24.00
Circle	100	1.60	99.60	4.60	68.20	17.00
Rectangle	100	0.74	99.80	5.70	75.60	16.00
Semicircle	100	4.80	96.80	24.00	78.40	49.00

% Succ: Percentage of successfully classified shapes. % A. Err: Percentage of non-coincident pixels with respect to the total area of the original figure.

Some results have been obtained testing the algorithm over sets of 2000 synthetic images, i.e., for each set, 500 triangles, 500 rectangles and so on were used for the evaluation purpose. Furthermore, 11 sets of figures with different levels of noise were generated, for σ ranging from 0 to 10 pixels. Besides, another seven sets of figures were also generated, each one with a different degree of occlusion, ranging from 10% of the size of the object to 40%.

In Table 1, numerical results for values of σ equal to 0 (non-noisy), 5 and 10 pixels are presented. For each level of noise, the success percentage and the localization accuracy are shown for the four shapes considered. Besides, in Fig. 11, we can see the performance of the algorithm as a function of the noise. In (a), the classification success percentage is shown for the four shapes considered. In (b), the accuracy in the localization of shapes as a function of noise is shown.

We can see that the success ratio is maintained even when the noise ratio is increased until around 7 pixels for σ. For higher values of noise, this ratio decreases quickly, specially for triangles. Regarding to localization accuracy, we can see that this value is quite acceptable for normal values of noise, but, even for non-noisy figures, there is about a 1-pixel error in the accuracy, mainly due to rounding errors. For semicircles, accuracy is normally worse than for the other shapes. The reason is that the comparison is over the whole ellipse that generates the semiellipse, while the ellipse is estimated with only half of the points available, since the other half belongs to the deleted part.

In Fig. 12, the performance of the algorithm in the presence of partial occlusions is shown. In (a), the classification success ratio as a function of the shape occlusion percentage is presented, while (b) shows the accuracy in shape localization. We can see that, up to 25% occlusions, both measures keep in a good performance, with an accuracy above 94% in classification and area error below 10%. For occlusions around 30% and above the

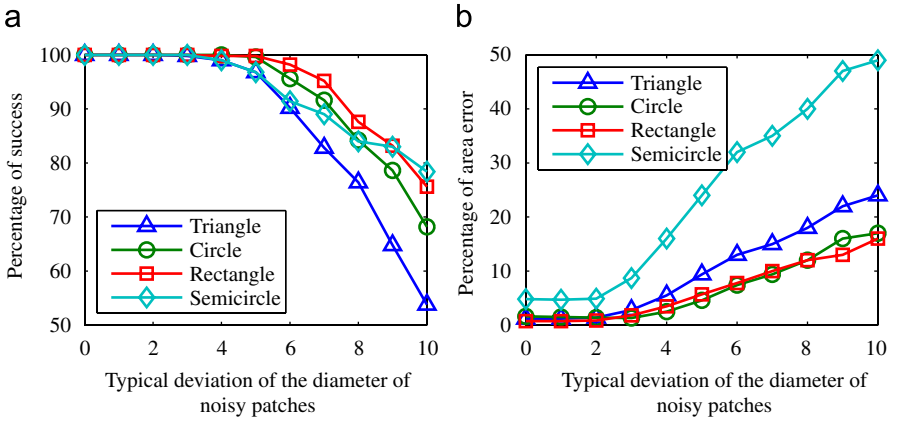

Fig. 11. Results as a function of noise.

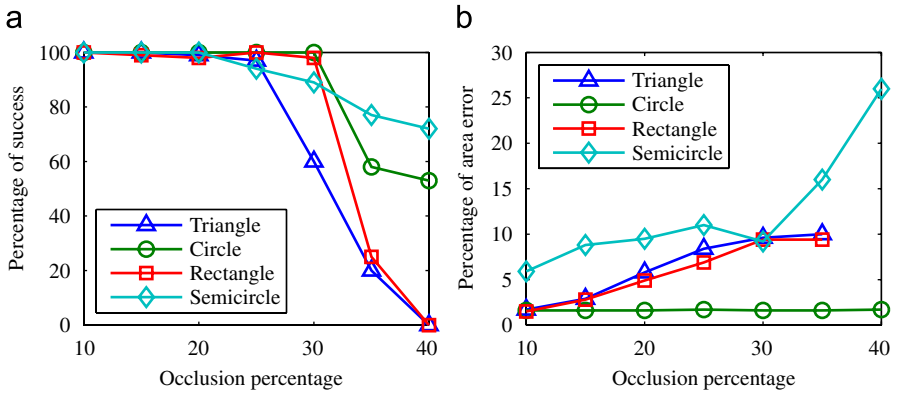

Fig. 12. Results as a function of the occlusion.

Fig. 13. Experimental results for the recognition step.

performance decreases rapidly because occlusions of such a high level implies that about one-quarter or more of the whole shape has disappeared, being this degradation more notorious for triangles and rectangles, which is inherent to the computer application designed for the construction of the figures. Since the application always occludes one of the vertex of the figure, this will always be more dangerous for the recognition than occluding any part of a circle. Even for occlusions above 35% the success probability is so low that we could not get valid data for the localization accuracy, as we can see in Fig. 12. For semicircles, we have the intermediate case.

The algorithm has also been prepared to return, as part of the results of its execution, the detected shape drawn over the original figure. The shapes are drawn with colour lines, as can be seen in Figs. 9 and 10. This utility has proven very useful in testing the algorithm on real images. In these examples, we can see the appropriate performance of the algorithm even in the presence of high levels of noise or occlusion, as illustrated in Figs. 9(b) or 10(b).

Finally, we would like to show the performance of the complete traffic sign recognition system developed by our group. Fig. 13 shows some images that include traffic signs. The system returns, as the result of its execution, the recognized signs drawn in the top-left corner of the figure, and the detected shape, for all the recognized signs, drawn over each sign. We can see that most of the signs had been successfully detected and classified. It can also be seen that some signs were recognized twice. This is due to the segmentation process. We use color segmentation when dealing with real images, so those signs composed of more than one color can be detected twice, one for each color.

6. Conclusions and future work

In this paper, we have presented a new algorithm for shape classification and localization of traffic signs. The main problems concerning this kind of system have been dealt with. The major advantage of the system is its good performance over different geometric distortions, like scaling, rotations and projective deformations. The algorithm has been tested with randomly generated synthetic images, using a test application implemented specially for this purpose. The evaluated parameters are the shape classification success probability, and the accuracy in the localization of the actual position of the sign.

The signature of the blob was used for the classification of the shape of the traffic sign. The normalization of the energy of the signature makes the algorithm invariant to image scaling, and the use of the absolute value of the FFT of the normalized signature makes the algorithm invariant to object rotations. Lastly, the computation of the homography between the object and a reference shape allows the algorithm to deal with camera projection distortions.

Future work will concentrate on the reduction of the misclassification probability, especially in the presence of high-level noise. In this sense, more complex classifiers, like support vector machines or neural networks can be used to improve the performance of the shape classifier.

Acknowledgments

This work was supported by the project of the Ministerio de Educación y Ciencia de España number TEC2004/03511/TCM and the project of the Comunidad de Madrid-UAH number CCG06-UAH/TIC0695.

References

[1] S. Maldonado, S. Lafuente, P. Gil, H. Gómez, F. López, Road-sign detection and recognition based on support vector machines, IEEE Trans. Intelligent Transportation Systems 8 (2) (2007) 264–278.
[2] A. de la Escalera, L.E. Moreno, Y. Hirai, M.A. Salichs, Neural traffic sign recognition for autonomous vehicles, in: IECON'94 20th International Conference on Industrial, Electronics, Control and Instrumentation, vol. 2, IEEE, New York, USA, 1994, pp. 841–846.
[3] A. de la Escalera, J.M. Armingol, M. Mata, Traffic sign recognition and analysis for intelligent vehicles, Image Vision Comput. 21 (2003) 247–258.
[4] A. de la Escalera, J.M. Armingol, J.M. Pastor, F.J. Rodríguez, Visual sign information extraction and identification by deformable models for intelligent vehicles, IEEE Trans. Intelligent Transportation Systems 15 (2) (2004) 57–68.
[5] S. Lafuente-Arroyo, P. García-Díaz, F.J. Acevedo-Rodríguez, P. Gil-Jiménez, S. Maldonado-Bascón, Traffic sign classification invariant to rotations using support vector machines, in: Proceedings of Advanced Concepts for Intelligent Vision Systems, Brussels, Belgium, 2004.
[6] P. Gil-Jiménez, S. Lafuente-Arroyo, S. Maldonado-Bascón, H. Gomez-Moreno, Shape classification algorithm using support vector machines for traffic sign recognition, in: Computational Intelligence and Bioinspired Systems, Barcelona, Spain, 2005, pp. 873–880.
[7] R. Janssen, W. Ritter, F. Stein, S. Ott, Hybrid approach for traffic sign recognition, in: IEEE International Conference on Intelligent Vehicles, IEEE, New York, 1993, pp. 390–397.
[8] W. Ritter, Traffic sign recognition in color image sequences, in: Intelligent Vehicles Symposium, IEEE, Detroit, USA, 1992, pp. 12–17.
[9] H. Liu, D. Liu, J. Xin, Real-time recognition of road traffic sign in motion image based on genetic algorithm, in: Proceedings of the 1st International Conference on Machine Learning and Cybernetics, 2002, pp. 83–86.
[10] S. Vitabile, G. Pollaccia, G. Pilato, F. Sorbello, Road signs recognition using a dynamic pixel aggregation technique in the hsv color space, in: Proceedings of the 11th International Conference on Image Analysis and Processing, IEEE, Palermo, Italy, 2001, pp. 572–577.
[11] D. Kang, N. Griswold, N. Kehtarnavaz, An invariant traffic sign recognition system based on sequential color processing and geometrical transformation, in: Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation, Dallas, TX, 1994, pp. 88–93.
[12] Y. Ishizuka, E.A. Puente, Segmentation of road sign symbols using opponent-color filters, in: ITSWC2004, Nagoya, 2004, p. 8.
[13] H. Sandoval, T. Hattori, S. Kitagawa, Y. Ghigusa, Angle-dependent edge detection for traffic signs recognition, in: Proceedings of the IEEE Intelligent Vehicles Symposium, IEEE, Dearbom, MI, USA, 2000, pp. 308–313.
[14] Y. Aoyagi, T. Asakura, A study on traffic sign recognition in scene image using genetic algorithms and neural networks, in: Proceedings of the 22nd IEEE International Conference on Industrial Electronics, Control and Instrumentation, vol. 3, Taipeh, Taiwan, 1996, pp. 1838–1843.
[15] P. Suau, Robust artificial landmark recognition using polar histograms, in: Lecture Notes in Computer Science, vol. 3803, Springer, Berlin, 2005, pp. 455–461.
[16] D. Gavrila, V. Philomin, Real-time object detection for SMART vehicles, in: Proceedings of IEEE International Conference on Computer Science, Kerkyra, Greece, 1999, pp. 87–93.
[17] G. Mo, Y. Aoky, Recognition of traffic signs in color images, in: TENCON 2004. 2004 IEEE Region 10 Conference B, 2004, pp. 100–103.
[18] S. Maldonado-Bascón, S. Lafuente-Arroyo, P. Gil Jiménez, H. Gómez-Moreno, F. López-Ferreras, Road-sign detection and recognition based on support vector machines, IEEE Trans. Intelligent Transportation Systems 8 (2) (2007) 264–278.
[19] P. Gil-Jiménez, S. Lafuente-Arroyo, H. Gomez-Moreno, F. López-Ferreras, S. Maldonado-Bascón, Traffic sign shape classification evaluation II: FFT applied to the signature of blobs, in: Proceedings of the IEEE Intelligent Vehicles Symposium, Las Vegas, USA, 2005, pp. 607–612.
[20] R. González, R. Woods, Digital Image Processing, Addison-Wesley, Reading, MA, 1993.
[21] A.K. Jain, Fundamentals of Digital Image Processing, Prentice-Hall, Englewood-Cliffs, NJ, 1989.
[22] R. Hartley, A. Zisermann, Multiple View Geometry in Computer Vision, second ed., Cambridge University Press, Cambridge, 2003.
[23] W. Gander, G.H. Golub, R. Strebel, Least-squares fitting of circles and ellipses, BIT 34 (1994) 558–578.