Noname manuscript No.
(will be inserted by the editor)

Embedded real time speed limit sign recognition using image processing and machine learning techniques

Samuel L. Gomes · Elizângela de S. Rebouças · Edson Cavalcanti Neto · João P. Papa · Victor H. C. de Albuquerque · P. P. Rebouças Filho · João Manuel R. S. Tavares

Received: date / Accepted: date

Abstract The number of traffic accidents in Brazil has reached alarming levels, and is currently one of the leading causes of death in the country. With the number of vehicles on the roads increasing rapidly, these problems will tend to worsen. Consequently, huge investments in resources to increase road safety will be required. The vertical R-19 system for optical character recognition of regulatory traffic signs (maximum speed limits) according to Brazilian standards developed in this work uses a camera positioned at the front of the vehicle, facing forward. This is so that images of traffic signs can be captured, enabling the use of image processing and analysis techniques for sign detection. This paper proposes the detection and recognition of speed limit signs based on a cascade of boosted classifiers working with haar-like features. The recognition of the sign detected is achieved based on the Optimum-path Forest classifier (OPF), Support Vector Machines (SVM), Multi-layer Perceptron (MLP), k-Nearest Neighbor (kNN), Extreme Learning Machine (ELM), Least Mean Squares (LMS), and Least Squares (LS) machine learning tech-

Samuel Luz Gomes, Elizângela de Souza Rebouças, Edson Cavalcanti Neto, Pedro P. Rebouças Filho
Laboratório de Processamento Digital de Imagens e Simulação Computacional, Instituto Federal de Federal de Educação, Ciência e Tecnologia do Ceará (IFCE), Ceará, Brazil E-mail: samuelluz@lapisco.ifce.edu.br, elizangelareboucas@lapisco.ifce.edu.br, edsoncavalcantineto@gmail.com, pedrosarf@ifce.edu.br

João Paulo Papa
Departamento de Ciência da Computação, Universidade Estadual Paulista, Bauru, São Paulo, Brazil. E-mail: papa@fc.unesp.br

Victor Hugo C. de Albuquerque
Programa de Pós-Graduação em Informática Aplicada, Laboratório de Bioinformática, Universidade de Fortaleza, Fortaleza-CE, Brazil. E-mail: victor.albuquerque@unifor.br

João Manuel R. S. Tavares
Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial, Departamento de Engenharia Mecânica, Faculdade de Engenharia, Universidade do Porto, Porto, Portugal. E-mail: tavares@fe.up.pt

niques. The SVM, OPF and kNN classifiers had average accuracies higher than 99.5%; the OPF classifier with a linear kernel took an average time of 87 μs to recognize a sign, while kNN took 11,721 μs and SVM 12,595 μs. This sign detection approach found and recognized successfully 11,320 road signs from a set of 12,520 images, leading to an overall accuracy of 90.41%. Analyzing the system globally recognition accuracy was 89.19%, as 11,167 road signs from a database with 12,520 signs were correctly recognized. The processing speed of the embedded system varied between 20 and 30 frames per second. Therefore, based on these results, the proposed system can be considered a promising tool with high commercial potential.

Keywords Cascade haar-like features · Pattern Recognition · Computer Vision · Automotive Applications.

1 Introduction

Car accidents are one of the major causes of death worldwide. Estimates made by the Secretary of Politics of Social Security, show that, in Brazil, the number of people permanently disabled as a consequence of traffic accidents increased from 33,000 to 352,000 between 2002 and 2012. In addition, the number of deaths increased from 46,000 to 60,000 in the same time period. Consequently, close to one million benefits paid nowadays by the National Social Security Institute (INSS) are for victims of car accidents. This cost represents more than 12 billion Brazilian Reais from INSS funds. The data from the Lider Insurance Company also indicates that most victims are between the ages of 18 and 40. The benefit that generates the greatest expense to the INSS is retirement due to disability, because it is a benefit that is paid for a long period of time, and in most cases to young people [1].

Given this scenario, manufacturers such as Volvo, Toyota and Ford are investing in technologies like Advanced Driver Assistance Systems (ADAS) in their vehicles to ensure the safety of occupants by helping to avoid potential accidents. The United States Department of Transportation estimates that an investment of US $ 1.2 billion in Intelligent Transportation Systems (ITS) technology would generate a return of US $ 30.2 billion in approximately 20 years. Likewise, Japan has been investing US $ 700 million annually in these technologies since 2004, while South Korea plans to invest US $ 3.2 billion between the years of 2008 and 2020 [2].

The goal of ADAS is to assist drivers, and consequently to significantly decrease the number of accidents. These systems use technologies such as: global positioning, radar, image sensors and techniques of computer vision. A study by the US Insurance Institute for Highway Safety estimated that the use of the intelligent assistance systems such as: lane departure warning (LDW), foreward collision warning (FCW), blind spot detection and adaptable headlights that are already available on the market can prevent or ameliorate one in three fatal collisions and one in five collisions that result in moderate or severe injuries [2].

Observing this trend, this work explores the use of computer vision as a solution for another major traffic safety problem: the driver's lack of attention to road signs, which causes a large number of accidents.

Many of the ADAS systems use computer vision techniques in their operations. For example, Lane Departure Warning System (LDWS) is a warning system that alerts the driver when he or she is veering out of or changing lanes by sending visual, audio and/or vibrational signals. This system is designed to minimize accidents by addressing the main causes, such as distraction and sleepiness [3, 4]. Adaptive cruise control systems read the speed limit signs through computer vision and alert the driver if he/she is not obeying the limit [5].

Assistance for vehicle parking uses ultra-sonic sensors and/or computer vision to indicate the proximity of objects. Current systems actively control the steering wheel, just leaving the driver with the control of moving the vehicle [6].

Other intelligent assistance systems are the Blind Spot Warning System (BSWS) and Sleepiness detector. BSWS is able to detect blind spots on the passenger side. This feature alerts the vehicle's driver if at the moment he/she is maneuvering there is any risk of collision [7]. Sleepiness detector is an intelligent system that monitors the driver's facial expressions to perceive the state of the driver's attention, alerting the driver that he or she should rest if signs of sleepiness or tiredness are detected [8].

Some car manufacturers have develop technologies, like an autonomous brake technology, that can stop a car when other vehicles or obstacles are very close and provides support to stay in the same lane, by applying a corrective force on the steering when the driver veers from the correct lane. A Cruise control adapter can also be used to automatically maintain a safe speed and a safe distance relatively to other vehicles, which in its most active form can prevent a driver from exceeding the speed limit.

Barthès and Bonnifait say that the development of Advanced Driver Assistance Systems to assist the driver and inform him/her of the road conditions can significantly contribute to reduce the number of accidents [9]. These systems should respond in real time to be useful and some of the major technologies used to ensure these requirements are: global positioning systems, image sensors and computer vision.

Cavalcanti Neto *et. al* [2] proposed a system to detect and recognize Brazilian vehicle license plates, in which the registered users have permission to enter a specific area. These authors used techniques of digital image processing were used, such as Hough Transform, Morphology, Threshold and Canny Edge Detector to extract the characters, as well as Least Squares, Least Mean Squares, Extreme Learning Machine, and Neural Network Multilayer Perceptron to identify the numbers and letters. Neto *et. al* [2] used motion detection to accelerate the embedded application previously developed because only the moving regions in the image were analyzed, which is not possible here because everything is moving in the input images.

Fig. 1: Illustration of the system developed in this work.

The main aim of this work was to develop an android application that can detect and recognize speed limit signs in real time. The system exhibits sign-detection in the car as shown in Figure 1. In order to develop the system, techniques of digital image processing (DIP) are used to extract, i.e. segment, the characters, along with techniques of machine learning (ML) to recognize the symbols obtained during the DIP stage.

This paper proposes detection of speed limit signs based on a cascade of boosted classifiers working with haar-like features in the DIP step. It also proposes the normalization of attributes for standardization of characters in the DIP stage, which is usually done in the pattern recognition step.

Among the contributions of this work for the pattern recognition step is the evaluation of seven classification methods and some of their variations in terms of suitability as an embedded application in real time. The recognition algorithms used were the k-Nearest Neighbors (kNN), Optimum-Path Forest classifier (OPF) configured with seven distance functions, Least Squares (LS), Least Mean Squares (LMS), Extreme Learning Machine (ELM), Artificial Neural Network Multilayer Perceptron (MLP) and Support Vector Machines (SVM) configured with four kernels. As far as the authors know, this is the first time that the OPF classifier has been analyzed in an embedded system.

2 Speed limit signs detection

The methodology proposed for the speed limit sign detection is based on a cascade of boosted classifiers working with haar-like features [10, 11]. This proposal will be compared to the methodology suggested by Neto *et al.* [2].

2.1 Based on a cascade of boosted classifiers working with haar-like features

Viola and Jones [10] proposed a rapid object detection algorithm using a boosted cascade of simple features, and Lienhart and Maydt [11] proposed an extended set of Haar-like features for rapid object detection.

This approach has been applied in various applications to detect objects in real time, such as face detection [12, 13, 14], pedestrian detection [15, 16], license plate detection [17], and object classification in microscopy [18]. However, as far as the authors know, this methodology has never been used for applications such as road sign detection neither incremented in an embedded solution.

The classifier used to detect speed limit signs is trained with a few samples of signs, called positive and negative examples [10, 11]. The positive examples included hundreds of images with speed signs, and negative examples included arbitrary images without valid signs. After a classifier is trained, it can be applied to a region of interest in an input image. The classifier outputs a "1" if the region is likely to show a sign, and "0" otherwise. To search for the object in the whole image the search window moves across the image and checks every location using the classifier. The classifier is designed so that it can easily find the objects of interest with different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image, the scan procedure is done several times using different scales.

2.2 Based on the Hough transform and Canny edge detector

This approach was proposed by Neto *et al.* [2] for the detection of license plates according to the Brazilian Standards. This approach uses the Canny operator combined with the Hough transform to detect objects.

The Canny edge detector performs two tasks: the filtering of noise and highlighting the pixels defining the border of an object in a digital image [19, 20]. To develop this algorithm, primary studies were focused on optimal borders, that can be represented by using functions in one dimension (1-D) [21, 22]. The authors showed that the best filter to start their algorithm was a smoothing algorithm, the Gaussian operator, followed by a Gaussian derivative, which in one dimension can be given by [23]:

$$d = (-\frac{x}{\sigma^2})e^{-\frac{x^2}{2\sigma^2}}, \tag{1}$$

where σ^2 consists of the data variance and x the input data. The Canny algorithm was designed to have three main properties: minimum error detection, good border locations, and minimal response time.

Edge detection has been used in many applications for object segmentation [24, 25, 26, 27], and the Canny detector has been commonly used to find objects, and Hough transform to recognize the right objects.

The Hough transform is a feature extraction technique used in digital image processing [28]. The aim of the technique is to find imperfect points on the object by comparing it to the desired object class through a voting process. The Hough transform algorithm elects in the voting process objects that have similarities to the desired shape.

The classic Hough transform was projected to identify lines on images, and it has been extended to identify other shapes, like circles or ellipses [28, 29]. Neto *et al.* [2, 30] applied it to detect lines, but in this work we will use it to detect circles.

3 Speed limit sign recognition

In this section, the seven machine learning techniques under comparison are introduced.

3.1 Support Vector Machines

One of the fundamental goals of the learning theory can be stated as: given two classes of known objects, assign one of them to a new unknown object. Thus, the objective in a two-class pattern recognition is to infer a function [31]:

$$f : \mathcal{X} \to \{\pm 1\}, \tag{2}$$

regarding the input-output of the training data.

Based on the principle of structural risk minimization [32], the SVM optimization process is aimed at establishing a separating function while accomplishing a trade-off between generalization and over-fitting.

Vapnik [32] considered a class of hyperplanes in some dot product space $\mathcal{H}$:

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0, \tag{3}$$

where $\mathbf{w}, \mathbf{x} \in \mathcal{H}, b \in \mathbb{R}$, corresponding to the decision function:

$$f(x) = \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b), \tag{4}$$

and, based on the following two arguments, the author proposed the Generalized Portrait learning algorithm for problems that are separable by hyperplanes:

1. Among all hyperplanes separating the data, there exists a unique optimal hyperplane distinguished by the maximum margin of separation between any training point and the hyperplane;
2. The over-fitting of the separating hyperplanes decreases with increasing margin.

Thus, to construct the optimal hyperplane, it is necessary to solve:

$$\underset{\mathbf{w}\in\mathcal{H},b\in\mathbb{R}}{\text{minimize}} \quad \tau(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2, \tag{5}$$

subject to :

$$y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1 \quad \text{for all} \quad i = 1, ..., m, \tag{6}$$

with the constraint (6) ensuring that $f(x_i)$ will be $+1$ for $y_i = +1$ and -1 for $y_i = -1$, and also fixing the scale of $\mathbf{w}$. A detailed discussion of these arguments is provided in [31].

The function τ in (5) is called the objective function, while in Equation 6 the functions are the *inequality constraints*. Together, they form a so-called *constrained optimization problem*. The separating function is then a weighted combination of elements of the training set. These elements are called *Support Vectors* and characterize the boundary between the two classes.

The replacement referred to as the *kernel trick* [31] is used to extend the concept of hyperplane classifiers to nonlinear support vector machines. However, even with the advantage of "kernelizing" the problem, the separating hyperplane may still not exist.

In order to allow some examples to violate Equation 6, the slack variables $\xi \geq 0$ are introduced [31], which leads to the constraints:

$$y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) \geq 1 - \xi_i \quad \text{for all} \quad i = 1, ..., m. \tag{7}$$

A classifier that generalizes efficiently is then found by controlling both the margin (through $||\mathbf{w}||$) and the sum of the slack variables $\sum_i \xi_i$. As a result, a possible accomplishment of such a *soft margin* classifier is obtained by minimizing the objective function:

$$\tau(\mathbf{w}, \xi) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \xi_i, \tag{8}$$

subject to the constraint in Equation 7, where the constant $C > 0$ determines the balance between over-fitting and generalization. Due to the tuning variable C, these kinds of SVM based classifiers are normally referred to as C-Support Vector Classifiers (C-SVC) [33].

The implementation used here for the SVM is the one suggested in [34] and [35].

3.2 Optimum-path Forest classifier

The Optimum-Path Forest (OPF) is a framework for the design of pattern classifiers based on optimal graph partitions [36, 37], in which each sample is represented as a node of a complete graph, and the arcs between them are weighted by the distance of their corresponding feature vectors. The idea behind OPF is to rule a competition process between some key samples (prototypes) in order to partition the graph into optimum-path trees (OPTs), which

will be rooted at each prototype. It is assumed that samples that belong to the same OPT are more strongly connected to their root (prototype) than to any other one in the optimum-path forest. Prototypes assign their costs for each node, and the prototype that offered the optimum path-cost will conquer that node, which will be marked with the label of the corresponding prototype.

Let $Z = Z_1 \cup Z_2$ be a dataset labeled with a function λ, in which Z_1 and Z_2 are, respectively, training and test sets such that Z_1 is used to train a given classifier and Z_2 is used to assess its accuracy. Let $S \subseteq Z_1$ be a set of prototype samples. Essentially, the OPF classifier creates a discrete optimal partition of the feature space such that any sample $s \in Z_2$ can be classified according to this partition. This partition is an optimum path forest (OPF) computed in $\Re^n$ by the Image Foresting Transform (IFT) algorithm [38].

The OPF algorithm may be used with any *smooth* path-cost function which can group samples with similar properties [38]. This work used the path-cost function f_{max}, which is computed as follows:

$$f_{max}(\langle s \rangle) = \begin{cases} 0 & \text{if } s \in S, \\ +\infty & \text{otherwise,} \end{cases}$$
$$f_{max}(\pi \cdot \langle s,t \rangle) = \max\{f_{max}(\pi), d(s,t)\}, \tag{9}$$

in which $d(s,t)$ means the distance between samples s and t, and a path π is defined as a sequence of adjacent samples. The $f_{max}(\pi)$ computes the maximum distance between adjacent samples in π, when π is not a trivial path.

The implementation used here for the OPF is the one suggested in [39, 40].

3.3 Least Squares

Least Squares (LS) was used first by [41], and is a very popular technique to make adjustments around a varied dataset:

$$y_k(i) = \varphi_k \left(\sum_{j=1}^{m} x_j \cdot w_{kj} \right). \tag{10}$$

From Equation 10, the output value from the network can be obtained through:

$$Y = W \cdot X, \tag{11}$$

where Y corresponds to the output matrix stimulated by the input vector X. Therefore, the W matrix is a matrix with a dimension of m x (k+1) because of the *bias* on the input system, that is: i=1,2...,m and j=1,2,...,k in Equation 10 [42].

Among the proposed LS models, this paper adopted the model proposed by [43]. The input attributes of the classifier are in each column of the X matrix and the vectors of the classifier outputs are in each column of the Y matrix.

In Equation 11, mathematical operations are used to achieve the goal, which is to isolate the W matrix. In order to remove the X matrix from the right side of the equation, it is necessary to multiply this side by the inverse matrix. However, the matrix must be square, so it is necessary to multiply by its transpose:

$$W = YX^T(XX^T)^{-1}. \tag{12}$$

The Optimal Linear Auto Associative Memory (OLAM) algorithm is used for both regression functions and classification. This classifier can be used either as a batch or iteratively depending on its application [44].

The implementation used here for the LS based classifier is the one suggested in [2, 1].

3.4 Least Mean Squares

According to [45], a Least-Mean Square (LMS) network is based on the use of instantaneous values, and the current values of the input network for the activation function [46]. The topology of the simple perceptron network is similar to the LS algorithm; however, they differ in their form of training. The output values are achieved as follows:

$$Y(i) = W \cdot X(i), \tag{13}$$

where *Y(i)* corresponds to the output matrix simple perceptron stimulated by the input vector *X(i)* and i corresponds to the actual iteration. Therefore, the W matrix is a matrix with the *m x (k+1)* dimensions due to bias from the input system where *i=1,2...,m* [42, 47].

The neuron activation function *y(t)* uses the signal function and the error value is calculated at each iteration. The W matrix is the weight matrix, which is obtained iteratively, using the derivative of the cost function:

$$w(t+1) = w(t) - \alpha \frac{\partial \xi(w)}{\partial w}, \tag{14}$$

where *w(t)* is the value of the weights from the previous iteration, and α is the learning rate [46].

Thus, the final result of W is obtained iteratively through the LMS matrix rule:

$$w(t+1) = w(t) + \alpha e(t)x(t). \tag{15}$$

The implementation used here for the LMS classifier is the one suggested in [2].

3.5 Extreme Learning Machine

The Extreme Learning Machine (ELM) is a neural network with a topology of a Single Hidden Layer Feedforward Neural Network (SLFN), which is a network that has a single hidden layer [48, 49].

The ELM uses a training method for its layers as follows: the weights of the hidden layer are randomly generated and the output layer weights are generated after the activation function of the hidden layer. The output of the hidden layer is used as input to the output layer and then the OLAM algorithm is used to obtain the values of the output layer weights [50].

The ELM algorithm, unlike other traditional algorithms, assumes a smaller training error and also the lowest standard of weights [50]. The disadvantage of the ELM is the need to use a high number of neurons in the hidden layer due to the need of higher hit rates, thus making the implementation of the algorithm in real-time embedded systems difficult, due to its high complexity and processing time [49, 51].

The matrix weights of the hidden layer w are generated randomly. After obtaining the weights randomly, it performs the activation of the neurons in the hidden layer from the input $x(t)$ of the system, thus obtaining the activated output of the hidden layer. The output of the hidden layer becomes the input of the output layer, thus transforming the network into a linear network [52].

The implementation used here for the ELM is the one proposed in [2].

3.6 Multi-layer Perceptron

The Multi-layer Perceptron (MLP) network is a Single-layer Neural Network (SLNN) organized in a cascade and subdivided in an input layer, one or more hidden layers and an output layer [53, 54, 45].

According to [55], SLNN does not represent separable functions linearly. This problem is solved by the use of two or more neurons with adaptive weights. However, it is necessary to use a training algorithm to adjust weights in these layers [56], to ones that perform error back propagation to compute the errors of hidden layers [45, 57].

One output layer with nonlinear neurons and one or more intermediate layers composed of neurons that represent the network activation function is the composition of a MLP network [58, 59, 60, 1]. The signal is always forward propagated, layer-by-layer.

The data for training was defined as follows: the input vector was equal to 1225, which is the result of 35x35 pixels size image vectorization, the class labels were 0-9 for the class of digits, as the number of neurons in the output layer is equal to the number of possible outputs, 10 digits.

In this work, we used a three-layer MLP with an input layer, a hidden layer and an output layer. According to [61], the classification of numbers on traffic signs can be made using a MLP network. Thus, we developed an MLP

network for the database used. The training of the MLP was based on the error backpropagation algorithm [45].

A network which has the number of hidden neurons equal to three times the number of classes (30 neurons) was used. The activation function used in the hidden layer was the logistic sigmoid function. Numbers were randomly generated between 0.0001 and -0.0001 for the initial weights of the network [62]. In the training of the network a decreasing learning rate was used with an initial value of 0.5.

For the stopping criteria of the network training, it was decided that the network should not be trained if the network spent 10 cycles without decreasing the mean square error or when this error was higher than the one of the previous epoch. The problem of these stopping criteria is that if the solution started to climb to a local minimum, the network training could continue. With that, we defined several starting solutions for the training to be sure that the network stops at the global minimum.

The implementation used here for the MLP was the one described in [63] and [2].

4 Proposed Framework

The framework developed in this work was built using C language for the Android Operating System (OS). The integrated algorithm for the automatic analysis of speed limit signs is composed of two main steps: Detection and Recognition, where the first step is to find the desired sign in an image with several objects, and the second step is to interpret the information on the sign, i.e. the maximum speed limit, Figure 2.

The first step in the computational pipeline developed is the speed limit sign detection via a cascade of boosted classifiers working with haar-like features [10, 11].

The classifier used to detect speed limit signs is trained with a few sample signs, called positive and negative examples [10, 11]. The positive examples included about 1,000 images with speed signs, and negative examples used were arbitrary images without validate speed limit signs. After a classifier is trained, it can be applied to a region of interest in an input image. The classifier outputs a "1" if the region is likely to show a speed limit sign and "0" otherwise. To search for the object in the whole image, the search window moves across the image and checks every location using the classifier. The classifier is designed so that it can easily find the objects of interest with different sizes, which is more efficient than resizing the input image itself. So, to find an object of an unknown size in the image, the scan procedure should be done several times using different scales.

The second step of the pipeline is the segmentation and identification of the digits, but before the segmentation, it is necessary to perform a thresholding of the sign for easy identification of the contours. These are identified through the application of an adaptive thresholding algorithm [64].

The next step is to filter the contours found in order to find the digits of the speed sign. The digits, before being sent to the next step, go through 4 filtering processes: by height, by width and through the spacing between digits.

The process of filtering checks the height of the average height of all objects and all filters that are above or below average with a tolerance of 15%. After the filtering process by height, the next step is to filter by width. The width filtering algorithm works the same as the algorithm for height, but based on the width. After identifying the digits in possible circles, the region where they are is segmented to separate the digits correctly. Figure 2 shows the possible circles found in green and the region where the digits are in blue.

After validation of the sign, the position of the digits, which are separated and standardized before the last step, is the recognition of the digits. This pattern recognition process assumes white digits with dimensions of 35x35 pixels on a black background. The digits are resized to make the algorithm invariant to distance.

The scaling of the digits occurs primarily by resizing the height which should be 33 pixels. Then, the width is defined in proportion to the original. Each digit after resizing is placed centrally in relation to the width.

The standardized digits were then subject to the LS, LMS, ELM, MLP, kNN, SVM and OPF classifiers. The recognition performance of these classifiers was evaluated by accuracy and processing time.

5 Results and discussion

This work proposes an efficient and powerful embedded system to recognize speed limit signs, where the stages must have high accuracy and low processing time. This section presents the results of the digital image processing and pattern recognition steps to find the best method to use in each step.

5.1 Speed limit digits detection

The proposed method in the step of digital image processing for the detection of speed limit signs is based on a cascade of boosted classifiers working with haar-like features. Figure 3 shows examples of speed limit sign digits segmented by the developed approach, showing the stages involved from the input image to the size standardized digits.

The detection step starts with an image acquired by a smartphone camera, and the images obtained are satisfactory, because the image sensor used, presents low noise, good focus and good brightness adjustments, as can be seen in Figures 3(a), 3(b), 3(c) and 3(d).

Figures 3(e), 3(f), 3(g) and 3(h) show the results of the conversion of the color acquired images to grayscale images. After applying the cascade of boosted classifiers working with haar-like features, these images have several

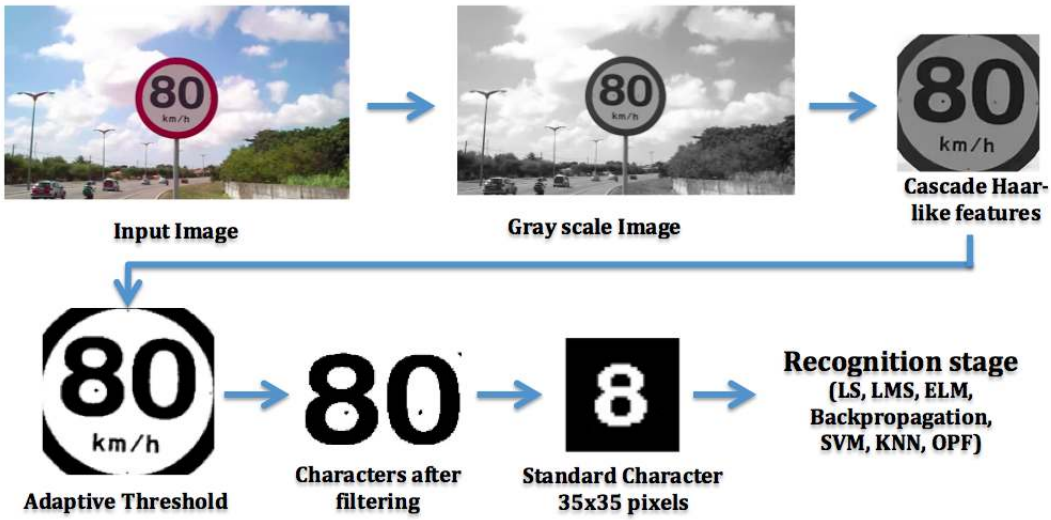

Fig. 2: Pipeline of the developed framework.

possible signs. After applying the adaptive thresholding (Figures 3(m) to 3(p)), the sign digits are obtained, and presented in Figures 3(u) to 3(x).

The test of the embedded system was performed using the speed limit signs of 20, 30, 40, 60 and 80 (km/h) as these are the most commonly used in urban environments. A total of 12,520 images were acquired with different inclinations and distances in streets and avenues of the city of Fortaleza and Maracanaú in the state of Ceará in Brazil.

From the $12,520$ images of the speed limit signs that were used as test, $11,320$ signs were properly located by the segmentation step, giving 90.41% of success in the detection of speed limit signs. On the other hand, the approach proposed by Neto *et al.* [2] based on the Canny operator combined with the Hough transform obtained only 45.3% correct results.

5.2 Speed limit digits recognition

In this work, we evaluated several classifiers to integrate an efficient and powerful embedded system to recognize the speed limit signs. Then, the recognition step using the k-Nearest Neighbors (kNN), Optimum-Path Forest (OPF), Least Squares (LS), Least Mean Squares (LMS), Extreme Learning Machine (ELM), Artificial Neural Network Multilayer Perceptron (MLP) and Support Vector Machines (SVM) based classifiers was carried out. The following section presents the results and discusses them.

From the results obtained in the step of digit detection presented in Section 5.1, a database with the segmented digits was built, Table 1. The feature

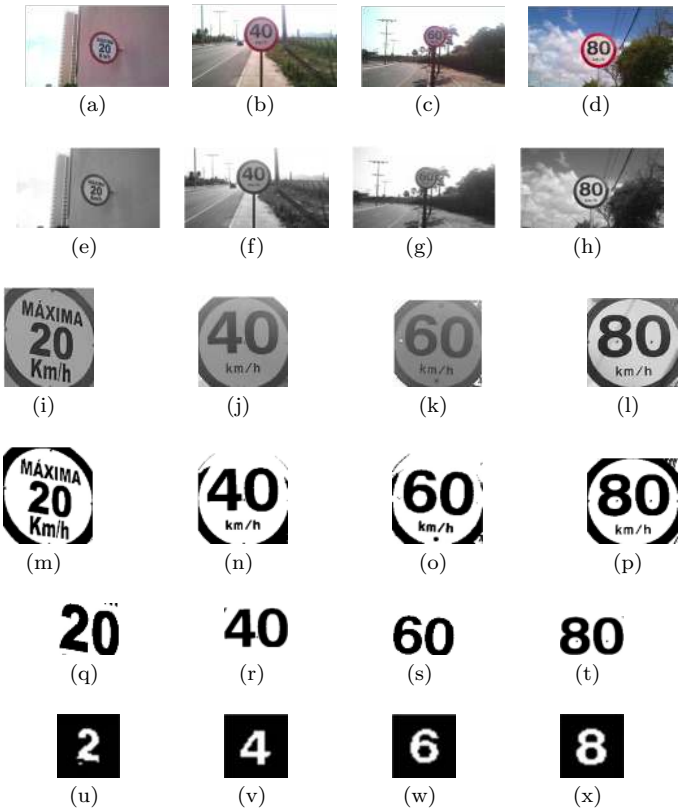

Fig. 3: Examples of results obtained for the segmentation of speed limit digits.

extraction approaches and classifier algorithms were combined to yield an intelligent system with high accuracy and low computational cost.

Table 1: Number of elements in each class of the speed limit sign digit database.

Digit	Class	N^o of Elements
0	1	1428
1	2	1841
2	3	1879
3	4	1688
4	5	1824
5	6	1569
6	7	1725
7	8	1414
8	9	1650
9	10	1952

For the training and test set sample sizes, a holdout procedure with 50% for the training and 50% for the test, with 10 steps, was employed. Each classifier was configured in various ways, and the best results obtained are the ones shown in Table 2. The kNN was configured with 1, 3 and 5 nearest neighbors. The SVM was configured using the linear kernels, polynomial, RBF and sigmoid, but only the linear and polynomial kernels had accuracy rates above 90%. The OPF was set with seven distances, but only the Euclidean and Chi Squared distance obtained accuracy rates above 99% for all samples. The MLP classifier trained by the ELM and MLP used 1,225 neurons in the input layer, 10 in the output layer and 30 in the hidden layer.

Each classifier was tested ten times, always shuffling the training and testing samples on a mobile device with Android OS 2.5 GHz Quad Core with 2GB of RAM.

Table 2: Results obtained in the evaluation of each classifier used for the recognition of the segmented speed limit digits. (Best values are in bold.)

Classifier	Maximum accuracy rate (%)	Minimum accuracy rate (%)	Mean accuracy rate (%)	Standard deviation	Average training time	Average testing time for a sample
LS	91.3	89.43	90.81	0.4	8.5 s	1.9 μs
LMS	90.80	45.00	76.45	17.52	50.56 s	4.1 μs
ELM	97.76	96.46	96.88	0.40	24.6 s	12.7 μs
MLP	96.83	87.12	91.93	3.08	158 s	14 μs
kNN (K=1)	99.83	98.51	99.14	0.61	0.027 s	10651 μs
kNN (K=3)	99.89	99.7	99.78	0.06	0.028 s	10639 μs
kNN (K=5)	99.79	99.71	99.76	0.03	0.029 s	11721 μs
OPF (Euclidean)	**99.65**	**99.36**	**99.54**	**0.10**	**2.5 s**	**87 μs**
OPF (ChiSquared)	99.7	99.23	99.47	0.13	2.5 s	748 μs
SVM (Polynomial)	99.88	97.04	99.43	0.94	70 s	9875 μs
SVM (Linear)	**99.87**	**99.76**	**99.82**	**0.04**	**40 s**	**5595 μs**

Table 2 summarizes the results obtained by each classifier. The values presented show that some classifiers were distinguished in terms of the training speed, mainly the ELM, kNN and SVM. Other classifiers stand out in terms of the speed to process a sample, such as the LS, LMS, MLP and SVM with linear kernel.

In terms of accuracy, the kNN, SVM and OPF classifiers were superior than the others, especially the kNN with 5 nearest neighbors, the SVM with linear Kernel and both OPF configurations. These classifiers are distinguished for the high average accuracies, always greater than 99%, presenting also high classification stability with low standard deviations.

Table 2 shows the accuracy rates and the prediction times to classify samples that are important criteria for embedded applications, and the OPF and SVM classifiers are the ones with the best results. To evaluate these classifiers further, Table 3 presents the *accuracy* (*Acc*), *sensitivity* (*Se*), *specificity*

Table 3: Acc, Se, Sp, FS for the worst case of the SVM, with linear and polynomial kernel, and OPF, with Euclidean and Chi Squared distances.

SVM

Linear kernel					Polynomial kernel				
class	Sp(%)	Se(%)	HM(%)	Acc(%)	class	Sp(%)	Se(%)	HM(%)	Acc(%)
0	99.94	100.0	99.95	99.72	0	99.97	100.0	99.97	99.86
1	99.96	100.0	99.96	99.83	1	100.0	74.04	97.18	85.09
2	99.97	100.0	99.97	99.89	2	100.0	100.0	100.0	100.0
3	100.0	99.17	99.91	99.58	3	96.87	99.88	97.17	87.53
4	99.94	99.78	99.92	99.67	4	100.0	99.78	99.97	99.89
5	99.98	100.0	99.98	99.93	5	99.96	100.0	99.96	99.80
6	99.98	99.76	99.96	99.82	6	100.0	99.53	99.95	99.76
7	99.97	99.85	99.96	99.78	7	99.98	99.85	99.97	99.85
8	99.97	99.51	99.92	99.63	8	99.97	99.63	99.94	99.69
9	99.98	99.59	99.94	99.74	9	99.94	99.89	99.94	99.74
Total	99.97	99.76	99.95	99.76	Total	99.67	97.04	99.40	97.04

OPF

Euclidean Distance					Chi Squared Distance				
class	Sp(%)	Se(%)	HM(%)	Acc(%)	class	Sp(%)	Se(%)	HM(%)	Acc(%)
0	99.93	100.0	99.94	99.71	0	99.97	99.81	99.96	99.81
1	99.95	100.0	99.96	99.80	1	99.97	99.81	99.96	99.81
2	99.61	100.0	99.65	98.37	2	99.95	99.81	99.94	99.72
3	99.95	96.48	99.61	98.02	3	99.97	98.97	99.87	99.39
4	100.0	99.64	99.96	99.82	4	99.93	99.81	99.92	99.62
5	100.0	99.81	99.98	99.90	5	99.91	99.62	99.89	99.44
6	99.95	98.91	99.85	99.27	6	99.65	99.08	99.60	98.00
7	99.97	99.82	99.96	99.82	7	99.97	100.0	99.98	99.91
8	99.91	99.07	99.83	99.16	8	99.79	96.51	99.47	97.31
9	99.95	99.82	99.94	99.73	9	99.95	98.89	99.85	99.25
Total	99.92	99.36	99.87	99.36	Total	99.91	99.23	99.84	99.23

(*Sp*) and *Harmonicmeans* (*HM*) metrics for each class under study for the worst case obtained.

The values presented in Table 2 show that the SVM with linear Kernel stands out as it has the highest accuracy and lowest standard deviation. Also, OPF with Euclidean distance had the lowest test time compared to the other classifiers; its training time was 16 times lower and the testing time was 64 times lower than the SVM with linear kernel. The OPF with Euclidean distance had an average accuracy of 99.54±0.10.

These findings confirm that the OPF classifier with Euclidian distance is suitable to be integrated in an android application for speed limit sign recognition with high efficiency.

The standardized digit sizes obtained in the DIP step, and used in the classifiers evaluation for the pattern recognition step are available at **website**

5.3 Overall results and main contributions

Many methods have been proposed to detect speed signs, often using some reference object in the input images. The first contribution of the framework proposed is the detection of speed signs based on a cascade of boosted classifiers combined with haar-like features. This approach detects speed signs independent of the image acquisition distance, which is an important feature since the signs are smaller the further they are from the camera. This is because samples of signs with various sizes were used in the training of the cascade classifier.

Another important contribution of the proposed framework is not having to use additional attributes, since the digits are resized to a standard size of 35x35 pixels. By doing this, it was found that the digits are invariant in terms of size, but here this is attained by processing the image and not in the recognition step as is normally done. This increases the recognition speed and robustness. Another contribution is also related to the processing speed, verifying seven types of classifiers to check which one had the best recognition performance and low processing time. The top recognition rate obtained was superior to 99.7% with the SVM and OPF classifiers performing in real time in the embedded system.

Analyzing the framework in an optimal configuration, we obtained a detection and recognition of 89.19%, which corresponds to 11,167 signs correctly detected and recognized from a database with 12,520 signs. The speed of the embedded system varied between 20 and 30 frames per second, depending on the number of signs found in the input image.

All the solutions that were developed here are fast and able to be embedded in commercial systems.

The drawback of the methodology developed is the error generated by large rotations, but this can be mitigated with the correct configuration of the camera.

6 Conclusion

The addresed problem is very challenging because with the growth of cities and the rise in the number of cars on the streets, it is extremely important to use systems able to identifying speed limit signs.

The objectives defined for this work were fully met, since the system developed is able to detect and recognize the speed limit signs satisfactorily. The developed system successfully segmented the speed signs and recognized their values with high accuracy.

The system obtained a global detection and recognition rate of 89.19%, with 90.41% in the detection step and 98.64% in the recognition step. From the images tested, it can be concluded that the system implemented is quite tolerant relative to the size of the image to be evaluated.

Even with good results, this work has some limitations. First, the system implemented is limited in terms of the distance between the image device and the speed limit sign, and the rotation involved, since when these are high, the identification can be erroneous.

Acknowledgements

Pedro Pedrosa Rebouças Filho acknowledges the sponsorship from the Instituto Federal do Ceará (IFCE) via grants PROINFRA/2013, PROAPP/2014 and PROINFRA/2015. Victor Hugo C. Albuquerque acknowledges the sponsorship from the Brazilian National Council for Research and Development (CNPq) through grants 470501/2013-8 and 301928/2014-2.

Authors gratefully acknowledge the funding of Project NORTE-01-0145-FEDER- 000022 - SciTech - Science and Technology for Competitive and Sustainable Industries, co-financed by Programa "Operacional Regional do Norte (NORTE2020)", through "Fundo Europeu de Desenvolvimento Regional (FEDER)".

References

1. S.L. Gomes, E.S. Rebouças, P.P. Rebouças Filho, Revista SODEBRAS **9**, 9 (2014)
2. E.C. Neto, S.L. Gomes, P.P.R. Filho, V.H.C. de Albuquerque, Measurement **70**, 36 (2015)
3. C. Tu, B. van Wyk, Y. Hamam, K. Djouani, S. Du, in *International Conference on Electronic Engineering and Computer Science (EECS 2013)*, vol. 4 (2013), vol. 4, pp. 316 – 322
4. S.C. Yi, Y.C. Chen, C.H. Chang, Computers and Electrical Engineering **42**, 23 (2015)
5. S. Yu, Z. Shi, Physica A: Statistical Mechanics and its Applications **428**, 206 (2015)
6. S. Carrese, S. Mantovani, M. Nigro, Transportation Research Part E: Logistics and Transportation Review **65**(0), 35 (2014)
7. B.F. Wu, H.Y. Huang, C.J. Chen, Y.H. Chen, C.W. Chang, Y.L. Chen, Computers and Electrical Engineering **39**(3), 846 (2013)
8. I. Garcia, S. Bronte, L. Bergasa, J. Almazan, J. Yebes, in *Intelligent Vehicles Symposium (IV)* (2012), pp. 618–623
9. J.P.A. Barthès, P. Bonnifait, in *Advances in Artificial Transportation Systems and Simulation* (Academic Press, Boston, 2015), pp. 163 – 180
10. P. Viola, M. Jones, in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1 (2001), vol. 1, pp. 511–518
11. R. Lienhart, J. Maydt, in *International Conference on Image Processing*, vol. 1 (2002), vol. 1, pp. I–900–I–903 vol.1

12. M. Rezaei, H. Ziaei Nafchi, S. Morales, in *Image and Video Technology*, *Lecture Notes in Computer Science*, vol. 8333 (Springer Berlin Heidelberg, 2014), *Lecture Notes in Computer Science*, vol. 8333, pp. 302–313
13. P. Elmer, A. Lupp, S. Sprenger, R. Thaler, A. Uhl, in *Image Analysis*, *Lecture Notes in Computer Science*, vol. 9127, ed. by R.R. Paulsen, K.S. Pedersen (Springer International Publishing, 2015), *Lecture Notes in Computer Science*, vol. 9127, pp. 53–64
14. A.P. Mena, M. Bachiller Mayoral, E. Díaz-Lópe, in *Bioinspired Computation in Artificial Systems*, *Lecture Notes in Computer Science*, vol. 9108 (Springer International Publishing, 2015), *Lecture Notes in Computer Science*, vol. 9108, pp. 175–183
15. G. Rakate, S. Borhade, P. Jadhav, M. Shah, in *IEEE International Conference on Computational Intelligence Computing Research (ICCIC)* (2012), pp. 1–5
16. S. Zhang, C. Bauckhage, A. Cremers, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), pp. 947–954
17. K. Zheng, Y. Zhao, J. Gu, Q. Hu, in *IEEE International Symposium on Industrial Electronics (ISIE)* (2012), pp. 1502–1505
18. F. Amat, P. Keller, in *10th International Symposium on Biomedical Imaging (ISBI)* (2013), pp. 1194–1197
19. J.H. da Silva Felix, P.C. Cortez, P.P. Rebouças Filho, A.R. de Alexandria, R.C.S. Costa, M.A. Holanda, in *MICAI 2008: Advances in Artificial Intelligence* (Springer, 2008), pp. 957–964
20. P.P. Rebouças Filho, P.C. Cortez, J.H.d.S. Félix, T.d.S. Cavalcante, M.A. Holanda, Revista Brasileira de Engenharia Biomédica **29**(4), 363 (2013)
21. J. Canny, IEEE Transactions on Pattern Analysis and Machine Intelligence (6), 679 (1986)
22. J.M.R. Tavares, P.P. Rebouças Filho, T.D.S. Cavalcante, V.H.C. de Albuquerque, Journal of Testing and Evaluation **38**(1), 1 (2009)
23. A. McAndrew, *Introduction do digital image processing with matlab* (Thomson Learning, 2004)
24. V.H.C. Albuquerque, P.P. Rebouças Filho, T. da Silveira Cavalcanti, J.M.R.S. Tavares, Journal of Microscopy **240**(1), 50 (2010)
25. P.P. Rebouças Filho, P.C. Cortez, A.C. da Silva Barros, V.H.C. Albuquerque, Expert Systems with Applications **41**(17), 7707 (2014)
26. P.P. Rebouças Filho, F.D.L. Moreira, F.G. de Lima Xavier, S.L. Gomes, J.C. Santos, F.N.C. Freitas, R.G. Freitas, Materials **8**(7), 3864 (2015)
27. F.D.L. Moreira, M.N. Kleinberg, H.F. Arruda, F.N.C. Freitas, M.M.V. Parente, V.H.C. de Albuquerque, P.P. Rebouças Filho, Expert Systems with Applications **45**, 294 (2016)
28. R.O. Duda, P.E. Hart, Communications of the ACM **15**(1), 11 (1972)
29. H.K. Yuen, J. Illingworth, J. Kittler, Image and Vision Computing **7**(1), 31 (1989)
30. E.C. Neto, E.S. Rebouças, J.L. Moraes, S.L. Gomes, P.P. Rebouças Filho, IEEE Transactions on Latin America **13**, 272 (2015)
31. B. Schölkopf, A.J. Smola, *Learning with Kernels* (MIT Press, 2002)

32. V.N. Vapnik, IEEE Transactions on Neural Networks **10**(5), 988 (1999)
33. C. Cortes, V. Vapnik, Machine Learning **20**(3), 273 (1995)
34. C. Burges, Data Mining and Knowledge Discovery **2**(2), 121 (1998)
35. C.C. Chang, C.J. Lin, ACM Trans. Intell. Syst. Technol. **2**(3), 27:1 (2011)
36. J.P. Papa, A.X. Falcão, C.T.N. Suzuki, International Journal of Imaging Systems and Technology **19**(2), 120 (2009)
37. J.P. Papa, A.X. Falcão, V.H.C. de Albuquerque, J.M.R.S. Tavares, Pattern Recognition **45**(1), 512 (2012)
38. A.X. Falcão, J. Stolfi, R.A. Lotufo, IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(1), 19 (2004)
39. J.P. Papa, A.X. Falcao, C.T. Suzuki, International Journal of Imaging Systems and Technology **19**(2), 120 (2009)
40. V.H.C. Albuquerque, C.V. Barbosa, C.C. Silva, E.P. Moura, P.P. Rebouças Filho, J.P. Papa, J.M.R.S. Tavares, Sensors **15**(6), 12474 (2015)
41. J.A. Plucker, A. Esping. Human intelligence: Historical influences, current controversies, teaching resources. (2016). URL `http://www.intelltheory.com`
42. G. Bittencourt, *Artificial Intelligence - Tools and Theories*, 3rd edn. (Federal University of Santa Catarina, 2006)
43. T. Kohonen, *Self-organization and associative memory: 3rd edition* (Springer-Verlag New York, Inc., New York, NY, USA, 1989)
44. O. Helene, *Method of least squares* (Livraria da Física, 2006)
45. S.O. Haykin, *Neural Networks and Learning Machines* (Pearson Prentice Hall, 2008)
46. B. Widrow, Proceedings of the IEEE **78**, 1415 (1990)
47. B. Widrow, R. Winter, IEEE Computer **21**, 25 (1988)
48. P. Horata, S. Chiewchanwattana, K. Sunat, Neurocomputing **102**, 31 (2013)
49. A.L.B.P. Barros, G.A. Barreto, in *Brazilian Conference on Intelligent Systems* (Curitiba, PR, Brasil, 2012)
50. G.B. Huang, D.H. Wang, Y. Lan, International Journal of Machine Learning and Cybernetics **2**, 107 (2011)
51. G.B. Huang, Q.Y. Zhu, C.K. Siew, Neurocomputing **70**, 489 (2006)
52. G.B. Huang, L. Chen, C.K. Siew, IEEE Transactions on Neural Networks **17**, 879 (2006)
53. D.W. Ruck, S.K. Rogers, M. Kabrisky, M.E. Oxley, B.W. Suter, IEEE Transactions on Neural Networks **1**(4), 296 (1990)
54. S. Nissen, *Implementation of a Fast Artificial Neural Network Library (FANN)* (2003). Department of Computer Science University of Copenhagen (DIKU).
55. M. Minsky, S. Papert, *Perceptrons* (M.I.T Press, 1969)
56. F.M. de Azevedo, L.M. Brasil, R.C.L. de Oliveira, *Neural networks with applications Control and Expert Systems* (Visual Books, 2000)
57. C. Medeiros, G. Barreto, Neural Computing and Applications **22**(1), 71 (2013)

58. M.A. Arbib, *The handbook of Brain Theory and Neural Networks* (M.I.T Press, 2003)
59. G. Barreto, R. Frota, Pattern Analysis and Applications **16**(1), 83 (2013)
60. S.J. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach (3rd Edition)* (Prentice-Hall, 2009)
61. H.E. Kocer, K.K. Cevik, Procedia Computer Science **3**, 1033 (2011)
62. W. Schimidt, IEEE International Conference on Neural Networks **1**, 598 (1993)
63. M. Riedmiller, H. Braun, in *IEEE International Conference on Neural Networks* (1993), pp. 586–591 vol.1
64. C.A. Glasbey, CVGIP: Graphical Models and Image Processing **55**, 532 (1993)