VOL. 14, NO. 3, FEBRUARY2019 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

REAL TIME DETECTION AND GEOLOCALIZATION OF REGULATORY TRAFFIC SIGNS

José de Jesús Salgado Patrón, Oscar Roa Valbuena and Albeiro Cortés Cabezas
Department of Electronic Engineering, Surcolombiana University, Neiva, Colombia
E-Mail: josesalgadop@usco.edu.co

ABSTRACT

In this paper, an application capable of detecting, recognizing and geolocating real-time traffic regulatory signs is developed. OpenCV library modules are used as the cascade classifier to detect objects and the kNN machine learning algorithm (k-Nearest Neighbors or nearest k-neighbors), for the recognition of the digits of the traffic signs. The location of the signs is done by GPS. In this way, an application performance of 92% is achieved.

Keywords: detection, recognition, location, GPS, OpenCV, haar cascade classiffier, kNN.

INTRODUCTION

Computer vision allows putting into practice concepts from different areas such as color physics, optics, electronics, programming, computer systems, etc. Giving place to that this science is able to interpret the structure of a three-dimensional world from one or several two-dimensional images of that world [1].

OpenCV (Open Source Computer Vision Library) is a code library which has a great focus for work in real time and a wide range of tools for image processing [2]. Among the most used in this work are: Filter for noise reduction as the Gaussian Blur [3], for object detection the cascade classifier [4] and the kNN algorithm for character recognition [4].

The cascade classifier used in OpenCV is of the Haar type. For its correct operation, it is necessary to take positive samples (in this case, regulatory traffic signs) and negative samples (images that are not regulatory traffic signs) [5]. In this way, it can be trained a classifier capable of detecting the desired object and whose result is used by a machine learning method [6].

The kNN algorithm (k-nearest neighbors or nearest k-neighbors) is a machine learning method supported in OpenCV. This algorithm compares pixels to classify them accordingly [6].

DEVELOPMENT OF THE APPLICATION

The scheme of Figure-1 shows, at a general level, the structure on which the application of detection and recognition of regulatory traffic signs is based.

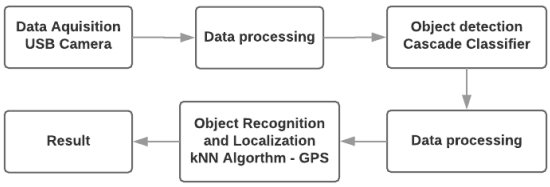

Figure-1. Application general scheme.

Data acquisition

For taking photos and videos, the USB camera has Right Light technology, which allows adapting to different brightness changes in the environment, allowing greater clarity. The videos are taken with a framerate of 30 frames per second for greater fluidity. Finally, the resolution with which the data is taken is 720p.

The initial data acquisition consists of a series of photos and videos taken on different streets and roads of regulatory traffic signs and everything that surrounds them, such as vehicles, pedestrians, buildings, trees, electric poles, etc. This material is used for the preparation of positive and negative samples and thus trains the cascade classifier.

Preparation of positive samples

It consists in the taking of photographs of regulatory traffic signs, in different conditions of brightness, contrast and angle of vision. As can be seen in Figure-2.

Figure-2. Positive samples untreated.

To use the package of positive samples, it is necessary to normalize the size of the images, highlight their main characteristics, reduce the noise and mitigate the effects of changes in lighting, as shown in Figure-3.

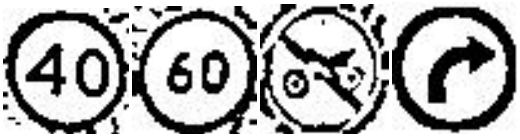

Figure-3. Positive samples treated.

Preparation of negative samples

The process for the negative samples is the same as for the positive samples, the difference being that the negative samples are everything that is not a regulatory traffic signal, for example, vehicles, poles, pedestrians, trees, buildings, etc.

Cascade classifier training

The classifier trained to meet the objective of the application had a total of 400 positive samples and 4000 negative images. The applications available in the OpenCV library, opencv_createsamples.exe and opencv_traincascade.exe [7], allow the classifier to be trained, the first application being in charge of preparing the positive samples and telling the classifier what to detect. The second application is the one in charge of automating learning, for this, it uses elements such as haar type filters, adaboost integral image and cascade classifiers [8].

kNN algorithm training

The result of the training this algorithm is to recognize the numbers from 0 to 9.

Class preparation

The classes are the numbers from 0 to 9. The preparation process consists of obtaining an image with all the digits repeated several times and each packet of digits with a different font type to increase the probabilities of recognition [6].

Preparation of tags

Tags are associated with each element of the class to define the response when the kNN algorithm designates which class the recognized object belongs to, that is, if there are 50 elements per class, 50 tags per class are needed and, in this way, to associate a response to each one of them [6].

Localization

To locate the detected and recognized regulatory traffic signals, a USB GPS is used. The obtaining of latitude and longitude data is achieved through the serial port emulated by the PC and read through the serial library of the Python programming language.

RESULTS AND DISCUSSIONS

Digital image processing

The objective of image processing is to eliminate noise in frames and the effects of lighting changes on objects. This makes it unnecessary to work in color space such as BGR or RGB [9]. To eliminate the noise the Gaussian filter is used and the effect of openning [6] and to mitigate the effects of the illumination is treated with the binarization of the image by means of the adaptive threshold method [6]. Figures 4 to 6show this process.

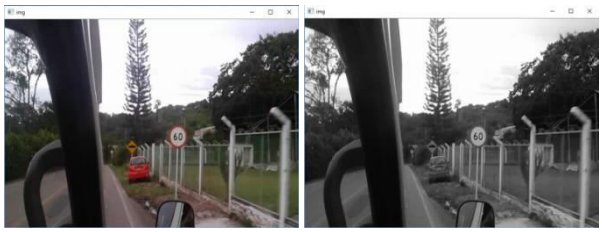

Figure-4. Original - Gray scale.

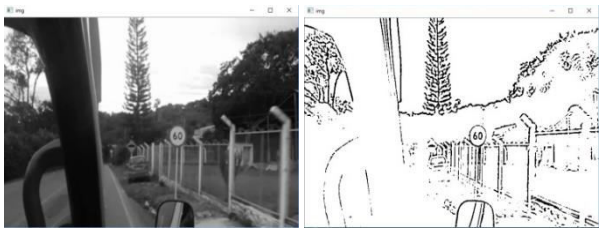

Figure-5. Gaussian filter - Threshold.

Figure-6. Erosion - Dilatation.

Under the result shown, a video test was used to test the classifier trained previously. The video has the following characteristics:

- Duration time: 1 minutes.
- 7 regulatory signals (6 speed, 1 forbidden to overtake)
- Areas of high vegetation.
- Multiple vehicles.
- Different structures (houses, meshes, bars, etc.)
- Other types of traffic signals.

The results can be seen in Table-1.

Table-1. Detection results with the cascade classifier.

Positives	False positives	Maximum detection distance	Minimum detection distance	Average speed
7	3	5m	1.5m	50km/h

In Figure-7 it can be seen that the classifier is able to detect regulatory traffic signals in different lighting conditions and at different distances.

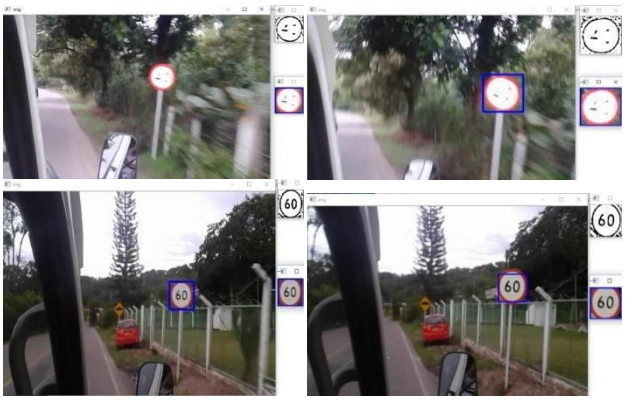

Figure-7. Detection with different lighting and distances.

Improvements in the detection of regulatory traffic signals

According to the results shown in Table-1, it is necessary to lower the number of false positives detected, affecting as little as possible the number of positives detected. To achieve a better performance, the detection of traffic signals is delimited by size and later validates the detection in the HSV color space [9], to verify that what the classifier detected in effect has the characteristic red color of the regulatory transit signals.

OpenCV treats all images as arrays. By means of this characteristic, a range is fixed in which the matrices whose rows have a size between 430 and 520 are valid. This implies that the distance to detect the signals is between 2m and 3.5m approximately.

The red ranges in the HSV color space are:

- Low reds: [0, 65, 75]
- High reds: [12, 255, 255]

It should be noted that OpenCV gives color values to pixels from 0 to 255.

These improvements are applied to the video test and equally to a second video.The characteristics of the second video are:

- Duration time: 5 minutes
- 12 traffic signs (9 speed, 3 forbidden to overtake)
- Areas of high vegetation
- Areas of low and high lighting
- Multiple vehicles, structures and other types of signals

The results, presented in Table-2, obtained after the detection improvements are quite good, taking into account the second test video, which is much more demanding, the percentage of success in the positive is 91.6% and the detection rate of false positives was reduced 87.5%.

Table-2. Detection results.

	Positives	False Positives
Video 1 with improvements	5	0
Video 2 without improvements	12	24
Video 1 with improvements	11	3

Recognition of regulatory traffic signals

In this section the kNN algorithm is integrated to recognize the regulatory traffic signals categorizing them as speed signs (giving their value) and non-speed signs. This algorithm was applied to the two test videos previously used, taking into account that in the first, 5 positive and 0 false positives signals are detected. In the second, 11 positive and 3 false positives signals are detected. The results can be seen in Table-3.

Table-3. Video recognition results.

Signal	Video 1	Video 2
60kph	OK	2 of 3
40kph	OK	OK
30kph	OK	OK
20kph	OK	OK
Not Overtake	OK	OK
False Positive	0	3
Success	100%	92.85%

In video 2, only one traffic signal is recognized incorrectly, and it is 60kph, recognized as 50kph, false positives are classified as non-speed signals.

According to the results of the two videos, the kNN algorithm has a great performance; this gives way to the last stage, which is the localization.

Localization

This step is the simplest, to get the latitude and longitude data every time a regulatory traffic signal is detected, the GPS class is imported into the main script. The result of the GPS class is a .kml file that can be located in Google Earth. The location hit rate is 100%.

Field operation

The road tests are based on three routes, therefore three videos (fragments of the total route) are analyzed with different lighting conditions, multiple and varied objects. The three analyzed videos have duration of 20 minutes.

The profile of the first video is:

- 2 signs of 20kph.
- 9 signs of 30kph.
- 7 signs of 40kph.
- 1 sign of 50kph.

- 7 signs of 60kph.
- 4 signs of 80kph.
- 15 signs of another type (not overtake; turn to the right, etc.)
- Average speed 55kph

The results of the performance can be observed in Tables 4, 5 and 6.

Table-4. Performance of the application - Route 1.

Total signs	**45**
Located	45
Detected	37
Recognized	28
Not detected	8
Not recognized	9
False Positives	2

Table-5. Causes of non-detection - Route 1.

Total not detected	**8**
Cause: Distance	5
Cause: Speed	3

Table-6. Causes of non-recognition - Route 1.

Total not recognized	**9**
Cause: Brightness	5
Cause: Speed	4

The profile of the second video is:

- 1 signal of 20kph.
- 10 signs of 30kph.
- 9 signs of 40kph.
- 6 signs of 60kph.
- 14 signs of 80kph.
- 10 other types of signs (not overtake; turn to the right, etc.)
- Average speed 50kph.

The performance result in this route is shown in Table-7.

Table-7. Performance of the application - Route 2.

Total signs	**50**
Located	50
Detected	48
Recognized	44
Not detected	2
Not recognized	4
False Positives	8

The profile of the third video is:

- 15 signs of 30kph.
- 7 signs of 40kph.
- 6 signs of 50kph.
- 9 signs of 60kph.
- 7 signs of 80kph.
- 9 other types of signs (forbidden to overtake, turn to the right, etc.)
- Average speed 60kph.

The performance result in this route is shown in Tables 8, 9 and 10.

Table-8. Performance of the application - Route 3.

Total signs	**53**
Located	53
Detected	47
Recognized	40
Not detected	5
Not recognized	7
False Positives	8

Table-9. Causes of non-detection - Route 3.

Total not detected	**5**
Cause: Distance	2
Cause: Speed	3

Table-10. Causes of non-recognition - Route 3.

Total not recognized	**7**
Cause: Brightness	4
Cause: Speed	3

The field tests, as can be seen in the different results tables, are satisfactory. Overall, the performance of the application is approximately 90%, although changes in lighting play an important role when detecting and recognizing the signs, the factors that most affected the drop in the performance of the application were distance and speed. The distance due to the range reduction to avoid the detection of false positives and the speed because it distorts the image.

The location of each of the detected signs is done in Google Earth, as is shown in Figures 8, 9 and 10.

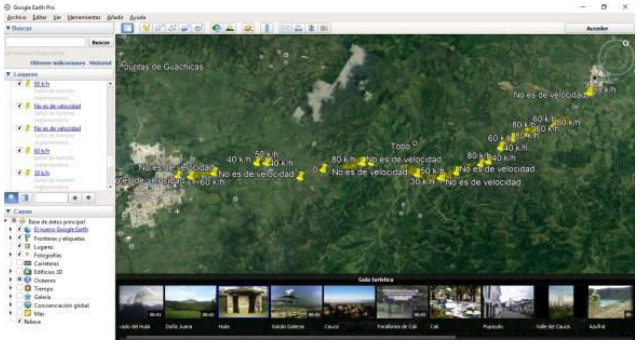

Figure-8. Signs location - Route 1.

Figure-9. Signs location - Route 2.

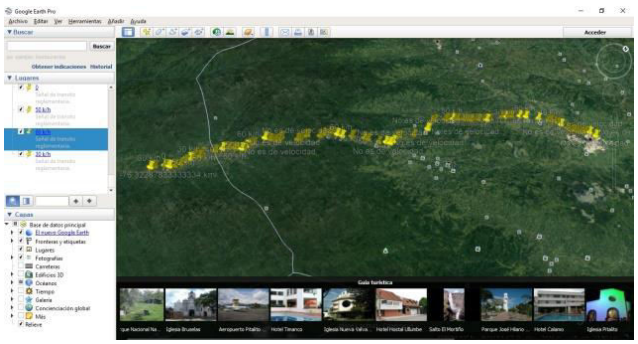

Figure-10. Signs location - Route 3.

CONCLUSIONS

The data preparation to train a cascade classifier or the kNN algorithm is the most important part of the process because the performance of these is affected by the quality of data available for this task.

The cascade classifier turns out to be a highly effective object detection method; the *adaboost* system allows dealing with large amounts of features saving computation time, which allows it to be used in real time applications.

In outdoor spaces the environment is really hostile for the detection and recognition of objects, especially by the strong changes of lighting that occur at all times. In order to solve this problem as much as possible, it is necessary to apply methods capable of adapting to these changes. This is the case of Adaptive Threshold method of OpenCV, itcan calculate a threshold each time the lighting changes.

The kNN algorithm is the most basic machine learning technique; however, it is effective when used in simple classifications, such as the case of digits. Its use for applications in real time, it turns out not to be the most convenient since it must calculate many characteristics until determining a result, thus taking a lot of processing time.

REFERENCES

[1] E. Alegre, G. Pajares y E. A, Conceptos y Métodos en Visión por Computador, Grupo de Visión del Comité Español de Automáticas, 2016.

[2] O. Team. «https://opencv.org/» [En línea]. Available: https://opencv.org

[3] O. d. Team. «https://docs.opencv.org,»Available: https://docs.opencv.org/2.4/doc/tutorials/imgproc/gausian_median_blur_bilateral_filter/gausian_median_blur_bilateral_filter.html.

[4] O. d. Team. «https://docs.opencv.org,». Available: https://docs.opencv.org/2.4.13.4/modules/ml/doc/k_nearest_neighbors.html.

[5] O. d. Team. «https://docs.opencv.org,». Available: https://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html.

[6] K. A. Mordvintsev y Abid, «https://readthedocs.org,». Available: https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf.

[7] B. O. 3. «http://docs.opencv.org.,»Available: http://docs.opencv.org/trunk/dc/d88/tutorial_traincascade.html.

[8] Jones, P. Viola y M. J, Robust Real-Time Face Detection, Netherlands: Kluwer, 2004.

[9] L. Enrique. Sucar y Giovani. Gómez, Visión Computacional, Puebla, 2011.