A PATTERN RECOGNITION SYSTEM FOR DETECTION OF ROAD SIGNS

Bochra TRIQUI, Abdelkader BENYETTOU

Center for Artificial Intelligent, USTO-MB University, Algeria

triqui_bouchra@yahoo.fr
a_benyettou@yahoo.fr

Abstract: Road sign identification in images is an important issue, especially for vehicle safety and road management applications. It is usually tackled in three stages: detection, recognition and tracking, and evaluated as a whole. To progress towards better algorithms, we focus in this article on the first stage of the process, namely road sign detection. We focus our work on a feature-based approach to build geometrical models of various kind of shapes: triangle, square, and circle form.

Keywords: Road Signs, Detection, Pattern Recognition.

Introduction

The purpose of this article is to develop a program that can detect or even recognize signs in a photo. The applications of this project are many: by equipping the camera cars and this program, we will be able not only to offer assistance to the driver (he will no longer need to search the panels since they will be displayed on a screen on the dashboard), but one could also consider a car autopilot, which would need to know the rules particular places where it rolls, and thus detect and recognize the panels.

The presented work in this paper proposes recognition system for detection of road signs, and is organized as follows:
In the first section, we introduced the problematic and then examined related works to classify method of recognition of road signs in Section 2. The third section presents details works .Our experimental results and conclusion are presented in Section 4.

Related Works

In the literature, we find several techniques applying various methods, the goal of which is to simplify the detecting of road signs; we have been interested in the following works:
A recent example of a learning-based detector is an attentive cascade of classifiers selected by Adaboost (Baro et al., 2009). The size of the input window of a detector is 30x30 (minimum size of detectable panels). This makes it possible to detect circular and triangular panels: four detectors are constructed, respectively modeling the danger panels, to give up Passage, Prohibition and Obligation. This approach gets the following results:
- Circle Prohibition: 70% detections for 3.34% false positives per image,
- Obligation circle: 60% detections for 0.82% false positives per image,
- Danger triangle: 65% detections for 2.21% false positives per image,
- triangle Give way: 75% detections for 2. 5% false positives by picture.

In (Ruta et al., 2009), the authors compare the detection of circular panels by transform from Hough to that obtained by a cascade of boosted classifiers and select the Hough transform for a geometric approach, the detection of panels is done from the outlines of the image. Thus, in (Fang et al., 2003) self-associating neural networks, whose weights are set to correspond to the desired form (circle, triangle, octagon), serve as nonlinear filters convolving the image at the level of of gray and the Hue canal. A fuzzy inference system exploits the two maps obtained to detect the panels. This process processes an image in 1 to 2 seconds, and is integrated into a tracking and recognition system, but the results are presented as illustrations only. Speed limit detection is also provided by a Hough transform by (Miura et al., 2000), the panels rectangular information being detected by a colorimetric thresholding in YUV followed by a horizontal and vertical projection of the gradient and a Kanji recognition. The performances of this approach are only illustrated by some examples.

 Copyright © The Online Journal of Science and Technology

In (Garcia-Garrido et al., 2006), they use a Hough transform to detect circles (speed limit or stop) or lines for triangles. Their approach selects closed contours in a certain width / height ratio range, which reduces robustness to out-of-plane rotations.

In (Piccioli et al., 1996), regions of interest are selected based on a thresholding in the Hue-Saturation HSV plane. Polygons are fitted to the linear contours chains, and assuming that the triangles are neither inclined nor distorted by perspective, equilateral triangles with a horizontal side and two slope sides are searched. This algorithm depends largely on the segmentation by the color, and it is strongly constrained in terms of orientation (it is restricted to the panels seen under a fronto-parallel perspective).

In the case of grayscale image sequences, (Barnes et al., 2003) use Radial Symmetry Transformation TSR with the same validation steps for the detection of 40 and 60 mph speed limit panels. (Caraffi et al., 2008) propose a detection of the signs of the end of speed limitation in grayscale image sequences. Their approach is to detect light / dark / light transitions: a Hough transform for circles validates successful candidates, and time filtering is required to eliminate many false alarms.

These filtering steps are not necessary in the approach we propose. According to our analysis, the TSR is a monovariate transformation where each contour point votes in several accumulators (one per scale) regardless of its neighborhood. This induces a relatively large number of false positives. In the generalized symmetry transformation TSG, introduced by Reisfeld (Reisfeld et al., 1995), each vote comes from a pair of points. This transformation is particularly suitable for highlighting radial and axial symmetries. It acts as a bivariate Hough transform, which reduces the number of false alarms and makes it less sensitive to noise than a single-crystal transformation.

Panel detection methods

The panel detection algorithms in a still image can be arranged according to the following three categories:
- colorimetric modeling: a related component segmentation based on a color model is performed. The regions of interest are then validated by a recognition algorithm or an appearance model. These methods are the fastest but also the least robust to variations in lighting conditions;
- geometric modeling: the contours of the image are analyzed by a structural or global approach. These methods are generally more robust than photometric ones because they process the gradient of the image, and can process grayscale images;
- methods with learning: a classifier (cascade, SVM, neural networks) is trained on the basis of examples. It is applied on a sliding window that traverses the image on several scales. These methods combine geometry and photometry but can be a costly step in computing time. They require
the constitution of a learning base by type of panels, tedious step when the number of objects to be detected is large.

Evaluation of algorithms

To evaluate our algorithm, a database of 700 photos was provided by the training supervisor. These photos are accompanied by data:
A solution mask, that is to say a binary image where the pixels corresponding to the panels are at 1 and the others at 0.

An annotation, that is to say a text file describing the position of the panels, their size, and their type.
These data will allow us to test our algorithm in many cases. In fact, this image database contains images where the panels are clearly visible, but also images where the panels are less visible (high inclination with respect to the camera, brightness too low, brightness too high, signs deteriorated, it contains even some signs that a man would have trouble noticing!).

We will consider the following:
- the true positives (noted TP): the places where the algorithm says there is a panel, and that there is actually one.
- false positives (denoted TN): the places where the algorithm says that there are no signs, and that there is not any.
- False Positives (FP): The places where the algorithm believes there is a panel, but there is not actually one.
- false negatives (FN): places where the algorithm believes there are no signs but there is actually one.

We will then consider:
- Precision: TP / (TP + FP).
- Accuracy: (TP + TN) / (TP + TN + FP + FN).

 Copyright © The Online Journal of Science and Technology

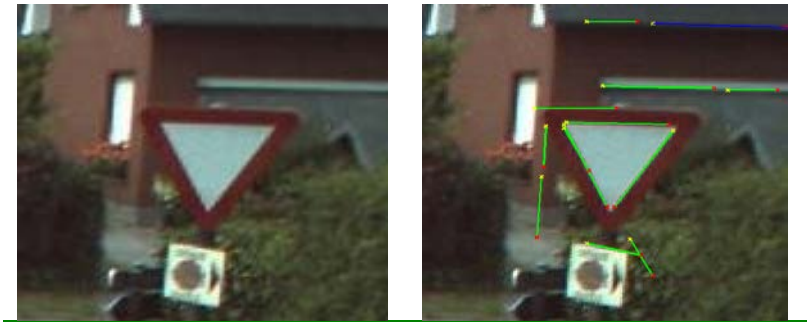
- The specificity: TN / (TN + FP)
- Sensitivity: TP / (TP + FN).

These values, as well as the duration of execution of each function will allow us to compare the different methods that we will test, in order to choose the best ones, and to know their advantages and disadvantages.
The results of the tests of the best functions will be given in annexes.

Search Triangles And Squares

Segment Search

The first step to finding triangles or squares, is to look for segments. For this, matlab offers functions that perform Hough transforms so that you can find segments (and indeed, the documentation is very well done).
We will therefore look for a dozen segments that these functions will suggest to us. They will return struct array whose attributes point1 and point2 can allow us to find these segments.



(a) Sign to detect (b) Segments detected

Figure 1: Segment detection

Once these segments are found, we will try to merge the points that are not too far from each other: it is likely that the detection of the segments is of poor quality, so we must expect that the segments detecting the panel edges do not touch each other perfectly.

The method that has been chosen is to separate the geometric information from the topological information.
We will build a table containing the location of points, and for each segment, we will test the proximity of its ends with the points that are already in the table. If the distance is small, we will consider that the two points are the same, otherwise we will add the current point to the table. We will then construct an array of integer pairs (that is, a matrix of size 2 × n) where two indices on the same column will mean that these indices form a segment. So we end up the arrival with a graph: these nodes and these arcs.

Search triangles
it is important to note that most triangular panels are either white panels with red edges or blue square panels containing a white triangle and a pictogram.

In any case, there will be a strong contrast between the colors at the borders of the triangles, which would not be the case if, for example, we had a red sign on a red background.

To find the triangles in an image, we will start looking for the paths of length 2 (that is to say containing 3 points) in the graph. So we will calculate the three angles of this triangle, and if each is worth 60° to 10°, we will retain the triangle. Assuming that the windows are well placed, we can suppose that in a window, there will be only one triangular panel, its area will be large and the triangle will be towards the middle of the window. As a result, only the triangle that has the best rating (we can for example note the triangles with a linear combination between the area and distance of the center of gravity of the triangle in the center of the window). We could also remove triangles that have no horizontal side.

 Copyright © The Online Journal of Science and Technology

Search rectangles
Unlike the previous case, the rectangular panels have no sharp edges, and therefore the detection of the segments that are the contours of the panels is significantly less effective. One can easily understand why by looking at this particular case of a parking sign :

(a) Image originale (b) Image de contours

Figure 2 : Difficult case

We see on the image of the outlines (calculated with the canny filter) that the edges of the panel are not visible. From there, the function of detection of matlab segments will not give any good, and a method like the one above will not work. It was still implemented (to the extent that this problem was identified after the tests). Several techniques have have been tried:
- search for parallel segments and creation of the quadrilateral
- search for perpendicular segments and creation of the quadrilateral

However, all these methods proved to be unsuccessful since the edges signs were poorly detected, it had to be particularly generous in accepting what we consider it as segments. From there, too much noise makes quadrilaterals are detected anywhere and whenever we have a window somewhere, we can be sure that this method will find a quadrilateral. This is hardly going to help us for the future. An idea to overcome this problem is to use a finer contour detector, but this has not been implemented.

Search circles
A circle detection function based on a Hough transformation was provided by the training supervisor. It has therefore been used on a smoothed contour image (since a circle viewed in perspective is no longer really a circle, so we have to leave a little room for this detection for have good results). It works well in many cases, but sometimes gives results for the least strange.

A proposal solution
One solution is to work on an outline image rather than an image that tells you about the colors. The idea was to follow these steps:
1. Calculate a contour image of the image
2. Calculate a distance map of contours
3. Drag a window (this time with a shape: triangular or circular ...)
We will keep the windows that minimize the sum of the pixels in the distance map, the idea being that since a panel is closed, the pixels inside the panel will be close to the edges and therefore of low value in the distance map.

Threshold used in our study

Dynamic thresholding

```
1 :function [red, blue] = Threshold1D(im, v1, v2)
2 :% Compute the red threshold
3 : red = im(:,:,1)./(im(:,:,2) + im(:,:,3)) > v1;
4 :% Compute the blue threshold
5 : blue = im(:,:,3) ./ im(:,:,1) > v2 & im(:,:,3) ./ im(:,:,2) > v2;
6 :% Dynamic threshold
7 :     if sum(red(:)) + sum(blue(:)) > numel(red(:))/25
8 :     [red,blue] = Threshold1(im,v1+0.1,v2+0.1);
9 :     end
10 :end
```

 Copyright © The Online Journal of Science and Technology

Technique	Average time per image	Precision	Accuracy	Specificity	Sensitivity
Threshold1	0.1364s	14.6791%	97.5053%	97.6732%	70.095 9%
Threshold2	0.1526s	9.0567%	95.4536 %	95.8943%	79.9986 %

However, it will be noticed that this method mixes red and blue colors. Indeed, if a red wall is detected, the blue threshold will be increased, and the detection of the blue panels will be degraded. So we did a second version of dynamic thresholding as well.

Table 1: Table of results with threshold1 and threshold2.

```
1: function [red, blue] = Threshold2D(im, v1, v2)
2 :% Compute the thresholds
3 :red = Red(im, v1);
4 :blue = Blue(im, v2);
5 :end
6 :function red = Red(im, v1)
7 :% Compute the red threshold of the image
8 :        red = im(:,:,1)./(im(:,:,2) + im(:,:,3)) > v1;
9 :        If there is more than 5% pixels that are white,
10 :       % increase the threshold
11 :       if sum(red(:)) / numel(red(:)) > 0.05
12 :       red = Red(im,v1+0.1);
13 :       end
14 :      end
15 : function blue = Blue(im,v2)
16 : % Compute the blue threshold of the image
17 :       blue = im(:,:,3) ./ im(:,:,1) > v2 & im(:,:,3) ./ im(:,:,2) > v2;
18 :       If there is more than 5% pixels that are white,
19 :       %increase the threshold
20:         if sum(blue(:)) / numel(blue(:)) > 0.05
21 :            blue = Blue(im, v2+0.1);
22 :        end
23 : end
```

Experiment Results

Threshold results

The threshold2 is the dynamic thresholding that simultaneously modifies the red and blue thresholding (same value for the threshold). The threshold1 is the one that treats red and blue separately.

Results for windows and shapes

We obtain these results with an average execution time of 2.1342s per image.

Table 2: Table of results.

shapes	Precision	Accuracy	Sensitivity
Windows	31.9978%	26.1959%	59.0956%
triangles	62.5551%	40.3409%	53.1835%
circles	12.6706%	8.6782%	21.5947%

 Copyright © The Online Journal of Science and Technology

Images results

The results images are available on this link. Red boxes represent enlarged windows (They have been enlarged to make sure that the entire panel is in the window). Red circles, and the cyan triangles represent the detected forms. The cyan rectangles are the smallest rectangles containing the detected shapes (for the triangles, they have been widened further since we suppose to have detected the inner border of the panel).

Conclusion

This work enabled us, on the one hand, to become even more familiar with the matlab language, and to review the techniques used in image processing and computer vision. It was also useful to see the difficulties commonly encountered in computer vision, namely the search for an algorithm that is effective in all conditions. Computer vision is a compelling topic that can be quite frustrating as it seems impossible to have an algorithm that delivers very good results.

References

Baro X., Escalera S., Vitria J., Pujol O., P.Radeva. (2009). Traffic Sign Recognition Using Evolutionary Adaboost Detection and Forest-ECOC Classification, *IEEE Transactions on Intelligent Transporation Systems*, vol. 10, (pp. 113-126).

Ruta A., Porikli F., Watanabe S., Li Y. (2009). In-vehicle camera traffic sign detection and recognition, *Machine Vision and Applications*.

Fang C., Chen S., Fuh C. (2003). Road sign detection and tracking, *IEEE Transactions on Vehicular Technology*, vol. 52, n° 5, (pp. 1329-1341).

Miura J., Tsuyoshi K., Shirai Y. (2000). An Active Vision System for Real-Time Traffic Sign Recognition, *Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC'00)*,(pp. 52-57).

Garcia-Garrido M., Sotelo M., Martin-Gorostiza E. (2006). Fast traffic sign detection and recognition under changing lighting conditions, *Proceedings of IEEE Intelligent Transportation Systems Conference (ITSC'06)*, Toronto, Canada, (pp. 811-816).

Piccioli G., Micheli E. D., Parodi P., Campani M. (1996). Robust method for road sign detection and recognition , *Image and Vision Computing*, vol. 14, n° 3, (pp. 209-223).

Barnes N., Zelinsky A., Fletcher L. (2003). Traffic sign recognition and analysis for intelligent vehicles , I*mage and Vision Computing*, vol. 21, n° 3,(pp. 247-258).

Caraffi C., Cardarelli E., Medici P., Porta P., Ghisio G., Monchiero G. (2008). An algorithm for Italian de-restriction signs detection , *IEEE Intelligent Vehicles Symposium*, (pp. 834-840).

Reisfeld D., Wolfson H., Yeshurun Y. (1995). Context Free Attentional Operators : the Generalized Symmetry Transform , International Journal of Computer Vision, vol. 14, n° 2, (pp. 119-130).

 Copyright © The Online Journal of Science and Technology