TRAFFIC SIGN DETECTION AND RECOGNITION

By

Sergi Sancho, Adriana Fernández, Eric López, Gerard Martí

Master in Computer Vision
Barcelona

Universitat Autònoma de Barcelona, Universitat Oberta de Catalunya, Universitat Politècnica de Catalunya, Universitat Pompeu Fabra

November 2015

Abstract

Recognition of road signs is a challenging problem that has engaged the attention of the Computer Vision community for more than 30 years. The first study of automated road sign recognition was reported in Japan in 1984. Since then, numerous methods have been developed for road sign detection and identification. The purpose of this document is to explain the advances and contributions done so far during the project 'Detection and Recognition of Traffic Signs I' (first module).

Keywords

Traffic sign detection, Color-based segmentation, *IHLS* segmentation, Mean-shift, *CCL*, Sliding window

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

The goal of this project is to apply some image processing techniques to a real world problem. In particular, the aim is to create a system able to detect traffic signs. A low level system will be modelled using features of the image as color, contour and shape to detect and classify objects in a scene. In this document, we are going to explain the different steps that we have done to obtain the final system.

To design a traffic sign detection and recognition system we have to consider some constrains:

- The traffic signs have different colors and shapes.
- The traffic signs can be in every place in the image.
- The traffic sign can be in various scales.
- Images have been taken with different lightning conditions, avoiding extreme lighting conditions as rain, snow...
- The viewpoint can be frontal or oblique.

The process of detecting traffic signs used in this project is the following:

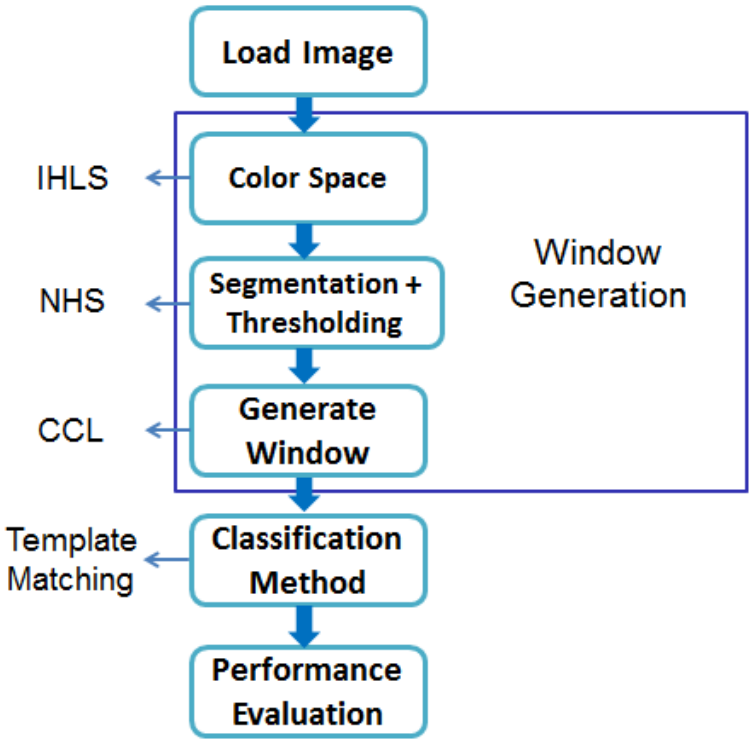


FIGURE 1.1: Traffic Signs Detection Process

2

Block 1 and 2: Window generation - Signal detection in color space

“ The aim of this block is to obtain a binary mask that contains all the objects that can be traffic signs. To generate a good binary mask we have studied different segmentation methods and we have applied appropriate thresholding criterion. ”

2.1 Characteristics of traffic sigs

This task was focus in have a first contact with properties of images that contains traffic signs. The traffic signs have a stipulated shapes and colors. Some of them are shown in 2.1

FIGURE 2.1: Example Traffic Signs

Our aim is to model a system able to detect if an input image contains a traffic sign. Consequently, we need a big data set to train the system. The data set used has been divided in two sets: a training set and a test set. We have used the training set to train our system and the test set to provide the evaluation performances. This data set contains the original images, the ground truth and a text file containing the locations of the signs. The first step is to save all the training set information provided in the

text files to know the characteristics of the traffic signs, as maximum and minimum size, aspect and filling ratio... We are going to use later this information to discard detected objects that are not traffic signs. In Table 2.1 is shown the characteristics of every type of sign we have considered.

Type	Max H	Min H	Max W	Min W	Max FR	Min FR	Max FilR	Min FilR	Frequency
A	253.39	32.29	238.13	43.34	1.34	0.81	0.49	0.29	0.19
B	245.98	30.58	213.79	29.75	1.50	0.84	0.72	0.43	0.14
C	239.20	37.35	250.25	36.80	1.54	0.93	0.77	0.53	0.11
D	201.47	29.46	183.83	30.96	1.54	0.91	0.76	0.00	0.14
E	198.30	36.08	345.76	37.77	1.74	0.94	0.97	0.47	0.20
F	244.53	44.59	325.78	45.93	2.21	0.70	0.98	0.00	0.18

Table 2.1: Characteristics of the signals. Where H = Height, W = Weight, FR = Form Ratio and FilR = Filling Ratio

2.2 Color Space and Segmentation

The initial approach to consider modelling a system is to find the color space that better represents our signs. Therefore, we have studied different color spaces as RGB, HSV, HSL and IHSL (Improved Hue Luminance Saturation). These color spaces should let us detect a differentiable characteristic of the signs: the color. Applying an appropriate threshold on the image we can obtain a first approximation of a binary mask that contains objects that have the specific color of the traffic signs (blue or red).

Our first step was to analyse the typical color spaces: RGB, HSV and HSL. After computing performance evaluation we obtain that the best color space was HSV. Finding the best threshold that segmented the images in HSV the results were not enough goods. For this reason, we looked for a new procedure. A segmentation method was applied to the images, the Mean Shift algorithm, and a color enhancement to the RGB color space.

2.2.1 First approach: Mean Shift ans RGB color enhancement

Mean Shift is a robust algorithm implemented by [Comaniciu and Meer, 2002] that segments the images into clear and unified color regions. In particular, it has three parameters that allow us to choose the characteristics of the regions (feature range bandwidth [hr], spatial bandwidth [hs] and minimum region area in pixels [M]). The parameters were fixed experimentally to better find full traffic signs. After computing segmentation we applied a color enhancement in RGB to the segmented image based on Tarik et al. [2013]. We try to highlight a certain channel of the images (in our case blue and red). Therefore, we compute the difference between the channel of interest and the others and we preserve if it has dominance over the others. We show an example of all the segmentation and thresholding process in Fig.2.2.

Figure 2.2: Mean Shift Segmentation + RGB Color Enhancement. 1. Original image, 2. Segmented image, 3. Binary mask

The color enhancement is provided for each RGB pixel x as it is shown in equations 2.1 and 2.2 and the binary mask containing the pixel candidates is obtained applying a threshold as 2.3 shows.

$$fR(x) = \frac{max(0, min(XR - XG, XR - XB))}{(XR + XG + XB)} \tag{2.1}$$

$$fR(x) = \frac{max(0, min(XB - XG, XB - XR))}{(XR + XG + XB)} \tag{2.2}$$

$$\begin{aligned} fR = fR > threshold, fB = fB > threshold \\ PixelCandidates = fR + fB \end{aligned} \tag{2.3}$$

The results after calculating performance evaluation with this procedure offered good results, although a huge computation time was needed (40 s/image).

2.2.2 Second approach: Improved Hue Luminance Saturation and Normalized Hue-Saturation method

The results using Mean Shift were good but we continue looking for a method that provides good results with a low computation cost. After looking for we found the color space IHLS and the segmentation method Normalized Hue-Saturation (NHS) specially performed for detect traffic signs in Manjare and Hambarde [2014]. The color space is changed to IHSL to provide more robustness with respect to changes in illumination. NHS segmentation is used to find the pixels that is likely to belong to traffic sign by color segmentation. In particular, NHS applies a thresholding approach to the input

Figure 2.3: IHLS color space + NHS segmentation method. 1. Original image, 2. Binary mask

image and returns a binary image representing the objects candidates to be traffic signs, as you can see in Fig. 2.3.

To improve the mask obtained in the previous task we apply morphological operators to complete or remove some objects in the masks. The interior of some signals are from a different color. In order to have a compact region we have to fill it. We have used MATLAB function imfill, with option "holes", to fill the regions that are completely surrounded by found pixels. Some of the holes were not completely surrounded, so we did an imclose first.

2.3 Results and Conclusions

In the following tables we will present the results obtained. Following the segmentation methods described before and all the system presented in 1, we show the pixel based evaluation and the window based evaluation over the test image data set:

Method	**Precision**	**Accuracy**	**Recall**	***F1* Measure**	**TP**	**FP**	**FN**	**tframe**
MS + CE	0.11	0.99	0.89	0.20	349010	412875	835332	10.65
IHLS + NHS	0.30	0.99	0.93	0.46	724035	1664308	49784	1.64

Table 2.2: Pixel based evaluation results.Where MS = Mean-Shift and CE = Color enhancement

Method	**Precision**	**Accuracy**	**Recall**	***F1* Measure**	**TP**	**FP**	**FN**
MS + CE	0.27	0.20	0.46	0.34	56	148	64
IHLS + NHS	0.71	0.62	0.83	0.77	118	46	24

Table 2.3: Window based evaluation results. Where MS = Mean-Shift and CE = Color enhancement

We can extract several conclusions from these results:

- We can discard to use Mean Shift segmentation because the computational time required is enough.

- IHSL + NHS is simpler to implement than Mean-Shift + RGB color enhancement.
- The best results are using the second approach, as you can see in Table 2.2 and Table 2.3. The precision and F1-measure using IHSL and NHS are 0.7. Therefore, the results are good enough to fix this method to the final system.

3

Block 3: Candidate window generation

"Once the image thresholding is performed, the next step is to generate the candidates. Our system implements *CCL* and *SW* methods."

3.1 Connected Component Labeling (*CCL*)

Given a structure element (4-connected or 8-connected), this technique works by scanning an image pixel-by-pixel in order to identify connected pixel regions, *i.e.* regions of adjacent pixels which share the same set of intensity values. After completing the scan, the equivalent label pairs are sorted into equivalence classes and a unique label is assigned to each class. As a final step, a second scan is made through the image, during which each label is replaced by the label assigned to its equivalence classes. Each final label represents a region [Ballard and Brown, 1982] (see Algorithm 1).

Once regions are found, the algorithm computes the bounding box for each one and use metrics to discard the regions that are not traffic signs. These metrics are the filling ratio, aspect ratio, and the shape measures.

This method is efficient and computationally inexpensive but some bounding boxes containing signals are discarded (*i.e.* due a bar on the signal, see FIGURE 3.1).

FIGURE 3.1: In this mask, using the *CCL* method, the bounding box is discarded due to the bar on the signal.

3.2 Sliding Window method (*SW*)

Sliding window method is based on a rectangular region (window) of a fixed width and height (sized for the average of the bounding boxes of the signals) that 'slides' across an image. For each of these windows, we would normally take the window region, compute the filling ratio and check whether it is inside the thresholds of filling ratios for signals (see FIGURE 3.2).

FIGURE 3.2: The left picture shows the input mask image and right picture shows the result of computing our *SW* method. As one can see, the major part of the noise is removed due its filling ratio and shape measurements. Thus, more than 15 candidates are discarted.

The same region can be detected by different windows, so we need to merge all the possible windows that detect the same region [Viola and Jones, 2004]. The windows

that detect the same region, due to the nature of our threshold, will be next to each other (they will form binary regions). Intuitively, the filling ratio should be at its highest in the center of that region, as we move the window away the filling ratio should diminish until it is out of the defined threshold. For this reason, to merge overlapped windows we find the centroid of each region, for that centroid we compute the corresponding window and we repeat it for all regions. This method is slower than *CCL*.

3.3 Conclusions

We propose two methods to generate candidates given the mask image. These are *CCL* and *SW*. No one is better than the other, each one has its own advantages and disadvantages. If our goal is to reduce the computation time, *CCL* will be the best choice.

In order to increase the system accuracy, we must filter the candidates to discard those that not fit with our traffic sign templates. Therefore, in the following section, template matching methods are detailed.

Algorithm 1 Two-pass *CCL*

```
1: Require Data 𝒟, Structure element 𝒮ℰ
2: function CCL(𝒟) return labels
3:    linked.init()
4:    labels.init(𝒟) {Initialize labels with background value}
5:    for each row r in 𝒟 {Assign temporary labels and record equivalences}
6:       for each column c in r
7:          if 𝒟[r][c] is not Background
8:             𝒩.getNeighbours(𝒮ℰ)
9:             if 𝒩 is empty
10:               linked[NextLabel] = set containing NextLabel
11:               labels[r][c] = NextLabel
12:               ++NextLabel
13:            else {Find the smallest value}
14:               Ln = Neighbors labels
15:               labels[r][c] = min(Ln)
16:               for each label in Ln
17:                  linked[label] = union(linked[label], Ln) {Join two subsets into a single
   subset}
18:               end for
19:            end if
20:         end if
21:      end for
22:   end for
23:   for each row r in 𝒟 {Replace each temporary label by the smallest label of its
   equivalence class}
24:      for each column c in r
25:         if 𝒟[r][c] is not Background
26:            labels[r][c] = find(labels[r][c])
27:         end if
28:      end for
29:   end for
30:   return labels
31: end function
```

4

Block 4: Window classification - Template matching

After performing a color segmentation and applying morphological operators to that segmentation, we have generated several candidate windows from the original image where the signal could be located. Now, we have to implement a method which will decide if that window contains a traffic signal or not. In future modules of the Master we will have to decide which type of signal we have detected, but for now we will focus on just deciding if a window contains a signal or not. In this chapter we will present our method of window classification based on template matching - We present a different approach with Geometry heuristics in chapter 5.

4.1 Introduction to template matching

Template matching is a form of shape detection that works by doing a comparison between our candidate and a set of matching masks which contain the shape that we are looking for. Applied to our project, the idea is to generate a set of templates with the shapes of all the possible signals and then compare them to each candidate window to see if we can find any of the shapes in them. The templates will be generated from the training data set provided for the project, and our method for generation is detailed in section 4.2. For comparison, we have proposed several methods: subtraction (section 4.3) shape of regions (using the chamfer algorithm, section 4.4) and correlation (section 4.4). Finally, the results and conclusions obtained are presented in section 4.5.

4.2 Generating the templates

We have to generate a template for each different shape present in our data set of traffic signals. From chapter 2 we already have made a division in four possible shapes: **circle**, **triangle**, **square** and **inverted triangle**. To create each of the shapes, we will use the training data set and the information we extracted from the data set in chapter 2. Our methodology is as follows:

- For each shape, take every mask corresponding to that shape.
- Crop the mask so that the resulting region is only the region where the signal is (That is, the region delimited as containing a signal in our data set).
- Resize the resulting image to a fixed size mask.
- Calculate the mean between all the resulting images (from one shape).

For the fixed size mask we use the maximum width from all regions that contains a signal from our dataset. We have a final size of $347x347$.

In figure 4.1 the final templates obtained are presented:

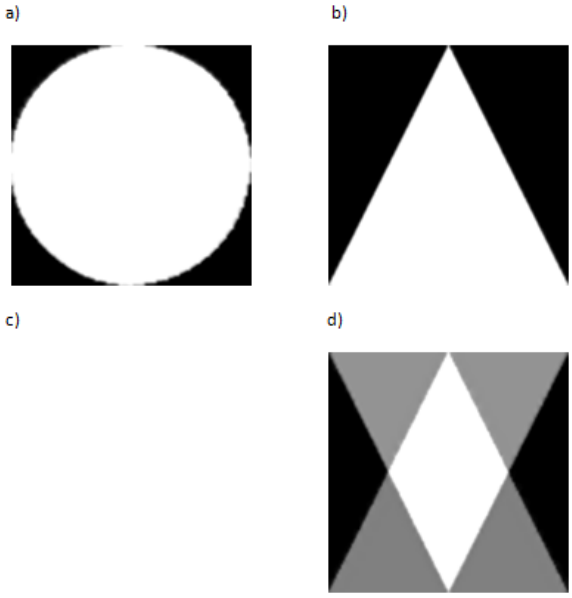


FIGURE 4.1: Final templates

As we can see in figure 4.1 a), our template for inverted triangles has some errors: we are using a normal triangle (or several) to compute our template. This is because of a mistake on the labeling in the data set, that labels a normal triangle as an inverted triangle, and leads to this kind of errors. Also, the template for squares is not missing: it is full white.

4.3 Template mathing using substraction

Once we have the templates, we can start building methods for comparison between the candidate windows and said templates. The first method we build is a rather simple subtraction. The basic idea behind is compute a subtraction between every candidate window and the template, and if the sum of the subtraction is less than a threshold, then it is accepted. Said threshold is found after extensive testing with the training data set to find the most appropriate value. This method is simple, but really fast.

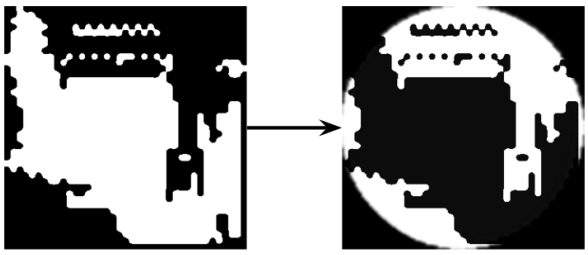


Figure 4.2: Example on applying the method on a candidate window.

4.4 Template matching using the shape of regions

This method of comparison is based on the chamfer distance model [H. G. Barrow, 1977]. A rough description of the algorithm used is explained below. For each candidate:

- Resize the candidate window to the template size.
- Compute the image contours of every template and the candidate window (to do this, we have used the canny algorithm).
- Calculate the distance transform (using MATLAB function bwdist()) with respect to the contours of the window candidate.
- Calculate the sum of the multiplication between the distance transform and every templates contour image.
- The window is candidate if the sum computed is less than a threshold.

As before, we have found the most appropriate threshold after computing a series of tests over the training set.

FIGURE 4.3: Example on applying the method on a candidate window.

4.5 Template matching using correlation

The last method we have implemented is a simple correlation method. To compare similarity between the template and the candidate window, we have used the MATLAB function *corr(A,B)*. Images that are similar to the templates are good candidates, and thus they will have a higher correlation. As in the two methods described before, we need an appropriate threshold to differentiate between good and bad candidates. To obtain the right threshold we operate in a similar way as before.

4.6 Results and conclusions

In the following tables we will present the results obtained. Following the segmentation described in chapter 2, for each type of candidate window generation and for each type of template matching, we show the pixel based evaluation and the window based evaluation over the test image data set:

Method	**Precision**	**Accuracy**	**Recall**	***F1* Measure**	**TP**	**FP**	**FN**	**tframe**
SlidingWin + Substraction	0.14818	0.99289	0.96748	0.257	353910	2034433	11895	49.90226
SlidingWin + Chamfer	0.04945	0.99202	0.81893	0.09328	118115	2270228	26116	44.52473
SlidingWin + Corr	0.09326	0.99240	0.91163	0.16922	222746	2165597	21592	45.52706
CCL + Substraction	0.31653	0.99404	0.90288	0.46873	755974	1632369	81319	0.56858
CCL + Chamfer	0.08206	0.99220	0.79281	0.14873	195998	2192345	51220	0.70453
CCL + Corr	0.30316	0.99404	0.93338	0.45766	724040	1664303	51680	0.58384

TABLE 4.1: Pixel based evaluation results.

Method	**Precision**	**Accuracy**	**Recall**	***F1* Measure**	**TP**	**FP**	**FN**
SlidingWin + Substraction	0.64286	0.64286	0.38571	0.48214	54	30	86
SlidingWin + Chamfer	0.42308	0.12941	0.15714	0.22917	22	30	118
SlidingWin + Corr	0.80357	0.29801	0.32143	0.45918	45	11	95
CCL + Substraction	0.69643	0.61257	0.61257	0.75974	117	51	23
CCL + Chamfer	0.51965	0.476	0.85	0.64499	119	110	21
CCL + Corr	0.72671	0.63587	0.83571	0.77741	117	44	23

TABLE 4.2: Window based evaluation results.

We can extract several conclusions from these results:

- Looking at the candidate window generation methods, CCL seems to provide the best results, while sliding window does not perform as good. Also, CCL is much

more computationally efficient. Improving the sliding window method by sliding over less possible windows could make the method go faster,

- Between the template matching methods, chamfer is the one that obtains poorer results. Simpler methods like subtraction and specially correlation obtain better results across the board.

- We see that good results in pixel based correlation does not always relate to good results in window-based evaluation, and viceversa. For this reason, we should always look to window-based evaluation results to get an appropriate picture of the performance of our system.

5

Block 5: Window classification - Geometry Heuristics

We will try to implement a method that takes into account directly the shape becaus the filling ratio approach may be too simple. We will use the Hough Transform in order to detect lines and use them to discard signal candidates that have not the correct shape.

5.1 Geometric Heuristics

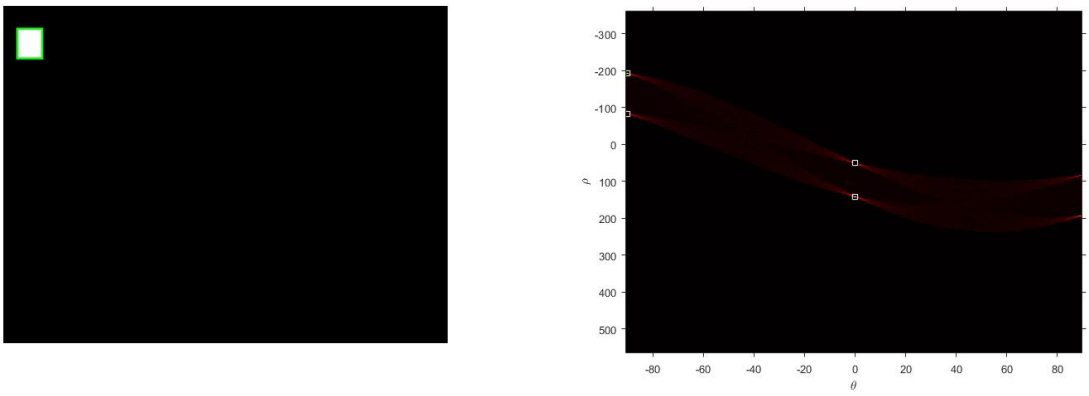
In order to solve the a detection problem we always have to take into account the characteristics of our detection target. As we have seen, we used the fact that traffic signals have bright colors in order to segmentate our images.

This time we will focus in another feature of the traffic signals. If we look to every the type of traffic signal we will realise that they all have simple shapes as triangles, squares or circles. Therefore, it will be very useful to have system that could distinguish regular figures from the strange ones.

This method will be the Hough Transform. Due to the nature of the method, in order perform the detection will use the countours of the figures. This will require to do a countour detection first like Canny Algorithm [Canny, 1986] or UCM [Arbelaez, 2006].

5.1.1 The Hough Transform

The Hough Transform was a technique proposed and patented in 1962, by Paul Hough [C, 1962]. Initially it was oriented to by a detection method of aligned points, but at the end it have been used to detect many types of figures.



(a) (b)

FIGURE 5.1: Example of a transformation from a rectangular figure. (a) Original image with the borders detected. (b) Hough transformed space with the four intersections marked.

The idea of this ingenous method is to use the equation of a straight line to transform every point of the real space into a straight line in the transformed space. This straight line will have as slope and intercept each component of the point in the real space.

Imagine we have a image with a straight line called r. The points (x_i, y_i) of this line will satisfy that:

$$y_i = a_0 + b_0 x_i \quad \forall x_i, y_i \in r \tag{5.1}$$

If we now perform the tranformation $(x_i, y_i) \rightarrow (a, b)$, so that every point will be transformed in a straight line such as:

$$a = -y_i + x_i b \quad \text{with } -\infty < b < \infty \tag{5.2}$$

So by (5.1) all these lines will intesect at a point (a_0, b_0) if the points of real were aligned, having effectively detected the straight line and computed is characteristics.

The problem of using this formulation will appear when the slope of the straight lines to detect tends to 0 or ∞ (horizontal and vertical lines). This aspect will be solved by a polar formulation where the space is transformed $(x_i, y_i) \rightarrow (\rho, \theta)$. The expresion wich will be used is:

$$\rho = x_i \sin\theta + y_i \cos\theta \tag{5.3}$$

Now the points will become sinusoids that will intesect in a point were the they were aligned. We can see the performance directly on FIGURE 5.1.

The good point of this technique will be that can be adapted to other types of shapes by choosing correctly the expresion of the transformation.

For example if we want to detect circles, we will need to transform every point of the 2-D image into a 3-D cone in the transformed space $(x_i, y_i) \rightarrow (a, b, r)$ using the next expression:

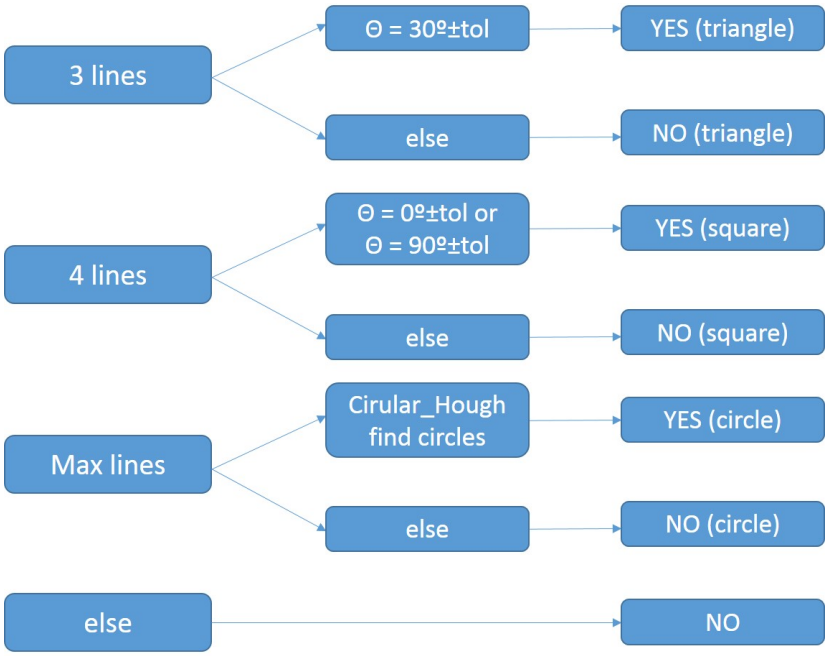


Figure 5.2: All the decision tree of the Hough method.

$$(x_i - a)^2 + (y_i - b)^2 = r^2 \tag{5.4}$$

Theoretically we would be able to perform this method to any kind of figure that can we discribe analytically. Nevertheless when we try to use it with a function which has too many parameters the complexity of the problem grows exponentially. This will limit very much the potential applications.

5.2 Implementation

Once we have understand how the Hough Transform works we will explain how we have used it in our signal detection system. The idea is to use the transform as a alternative of template maching and Chamfer methods.

Therefore, our input will be the window candidates generated with CCL or Sliding Window and the ouput will be *yes/no*.

The way to detect the intersections of the sinusoids will be by becoming the transformed space H in a 2-D histogram. Every time a curve go through a pixel will *vote* for this pixel. After all we will set a threshold (in our case will be $0,75 \cdot \max(H)$) in order to consider or not the existence of a straight line.

The scheme of the process will be:

1. Input: *Window Candidate* of the mask.

2. Perform a Canny Edge Detection in order to obtain the contours of the figures.

3. To apply the Matlab Hough transfrom function obtaining a voted space of posible lines.

4. Follow the scheme shown on FIGURE 5.2.

5. Output: *YES/NO*.

We wil perform all of this method over the CCL generation window method. The results we obtain can be seen on TABLE 5.1.

Method	**Precision**	**Accuracy**	**Recall**	***F1* Measure**	**TP**	**FP**	**FN**	**tframe**
Pixel Based	0.29222	0.99388	0.90940	0.44231	697910	1690433	69532	1.08705
Object Based	0.59016	0.49770	0.76056	0.66462	108	75	34	1.08705

TABLE 5.1: Results achieved with the Hough Transform Method.

5.3 Conclusions

After all the process we the F1measure we have obtained is $\approx 0,66462$ in the case of the object based evaluation technique. Although it is a very good mark if we take into account the simplicity of our method, it is not better than the results obtained before, using CCL+Correlation.

The reason of this result could be the difficulty of setting a good Hough detector. Morever, this method is not inmune to the outliers that can falsify our detector.

To sum up, we are pretty sataisfied with the results. Nevertheless, if we think in terms of complexity we would like to have achieved a better results than in the former section.

6 Conclusions

Based on the results we have achieved we can clearly choose the best combination of methods used during all the project.

It will be a color segmentation based on thresholds using **IHLS** that in terms of performance and behaviour has the best results. We will not use any morphological operator such as an opening or a closing due to the fact that we were not able to improve our evaluation marks with them. We will just use a Matlab *imfill* in order to complete the holes in our masks.

Regarding to the window generation block, we have achieved the best results by using **CCL** approach. We were able to make it perform very well comparing to the sliding window method. Especially if we pay attention to the performance time.

Finally as a window classification method we will use a template matching based on **correlation**. This is a very remarkable case. Because we may have though that the simplicity of the method will penalize the results comparing to oder mor complex method such as the Hough Transform Shape Detector. Nevertheless we were not able to achive a better behaviour with any other approach.

The final results of the final system can be seen on FIGURE 6.1. We can see that the F1measure of $\approx 0,77$ obtained is a very good result. It means that we detect more than the 83% of signals correctly.

Therefore, we are very satisfied with our method. Without using the most powerful approaches we were able to achieved a very good system overall.

Method	**Precision**	**Accuracy**	**Recall**	***F1* Measure**	**TP**	**FP**	**FN**	**tframe**
Pixel Based	0.30315	0.99404	0.93566	0.45794	724035	1664308	49784	1.10391
Object Based	0.71951	0.62766	0.83099	0.77124	118	46	24	1.10391

TABLE 6.1: Results achieved with the final system.

References

Pablo Arbelaez. Boundary extraction in natural images using ultrametric contour maps. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW'06. Conference on*, pages 182–182. IEEE, 2006.

D. Ballard and C. Brown. *Computer Vision*. Prentice-Hall, New Jersey, 1982. ISBN 9810224028.

H.P.V. C. Method and means for recognizing complex patterns, December 18 1962. URL http://www.google.com/patents/US3069654. US Patent 3,069,654.

John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5): 603–619, 2002.

R. C. Bolles H. C. Wolf H. G. Barrow, J. M. Tenenbaum. Parametric correspondence and chamfer matching: Two new techniques for image matching. *Proc. 5th Int. Joint Conf. Artificial Intelligence*, pages pages 659–663, 1977.

Pratiksha Manjare and SM Hambarde. Image segmentation and shape analysis for road sign detection. *International Journal of Engineering Research and Technology*, 3(12 (December-2014)), 2014.

Ayaou Tarik, Mourad Boussaid, Afdel Karim, and Amghar Abdellah. Improving road signs detection performance by combining the features of hough transform and texture. *International Journal of Computer Applications*, 73(9):5–9, 2013.

P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.