Road signs detection, classification and recognition

Luca Rinelli[1] Maicol Dolci[2] Tiago Abreu[3] Rui Alves[3]

[1]Politecnico di Torino, Italy
[2]Politecnico di Milano, Italy
[3]Facultade de Engenharia da Universidade do Porto, Portugal

Course of Sistemas Baseados em Visão
21st of December, 2018

Outline

Introduction

- Some Matlab scripts to work on a subset of the German Traffic Sign Detection Benchmark (GTSDB) and the German Traffic Sign Recognition Benchmark (GTSRB)
- All the code presented is already available open source on github[1]
- The work was divided in 3 different tasks: detection, classification and recognition

[1] *lucarinelli/Assignment-Sistemas-Baseados-em-Visao*. https://github.com/lucarinelli/Assignment-Sistemas-Baseados-em-Visao. Accessed: 2018-12-17.

Task 1: Experiments and challenges

Goal: detect road signs in a realistic road photo from the perspective of a car

Challenges:

- The images are very different among them and very often full of colored objects
- There is quite a wide range of different scenes and lights
- Some signs are very small, blurred or skew
- General pre-processing often led to worse results
- Lines and/or circles detection failing

The method used

The method used briefly consists of:

- HSV color segmentation and edges detection on gray-scale
- Masks operation in order to obtain different regions in the image
- Filter regions to make hypothesis
- Confirm hypothesis through classifier
  - Circles and lines detection
  - Shape masks and scoring
  - Verdict and ROI adjustment

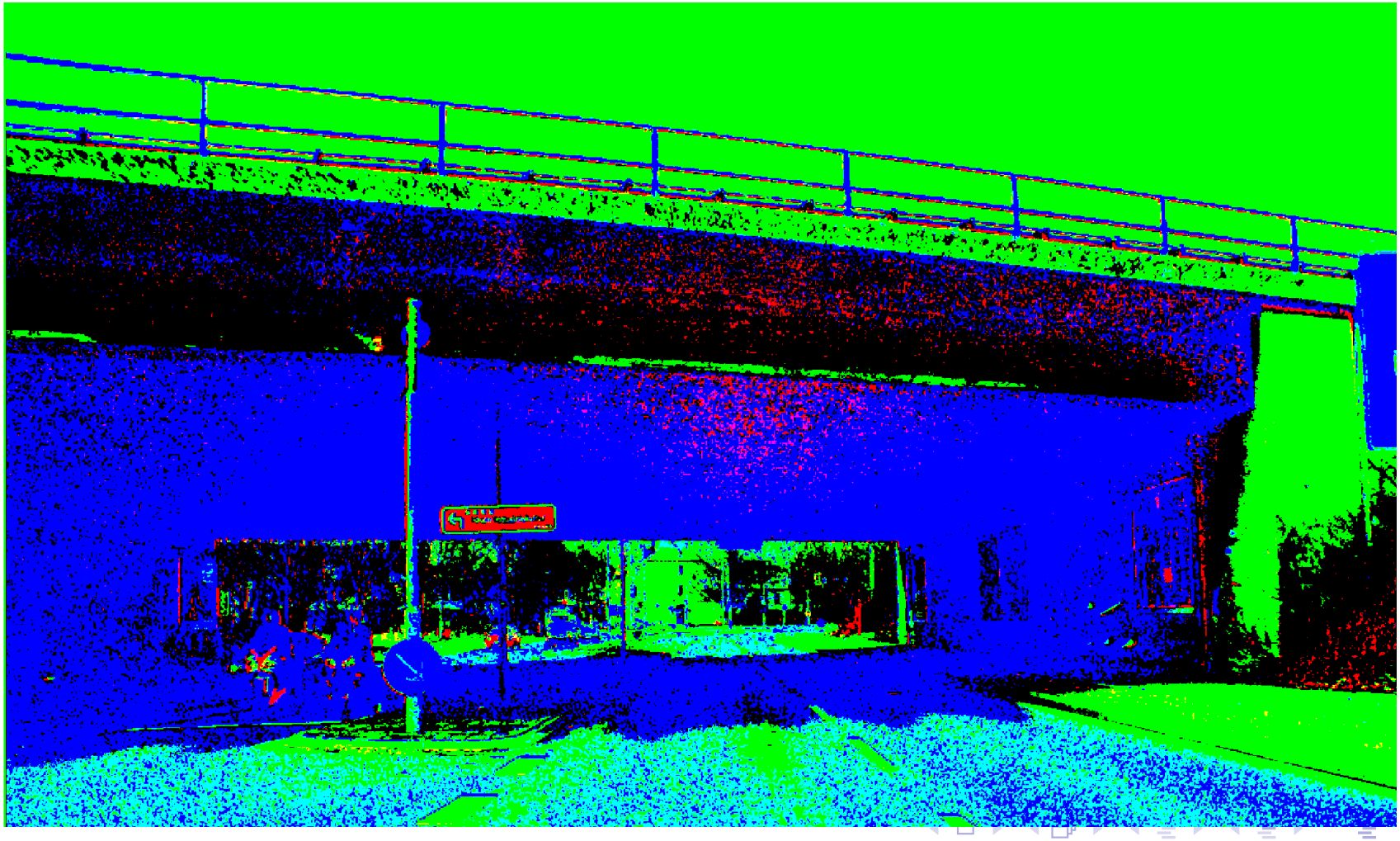
Masks

The first step of the approach used is the **color segmentation** of the image in **4 different binary images** and obtain the **edges** from a gray-scale version of the original image

```
1  just_red = createMaskRed(original);
2  just_blue = createMaskBlue(original);
3  just_whitish = createMaskWhitish(original);
4  just_yellow = createMaskYellow(original);
5
6  gray = rgb2gray(original);
7  edges = edge(gray,'canny');
```

The masks are obtained working on the **HSV** color space and are generated automatically using Matlab app for Color Thresholding

Putting all together (green component is the whitish mask)



Masks subtractions

```
pre_whitish = just_whitish;
old_red = just_red; old_blue = just_blue; % used later task2func
% remove blue from white
just_whitish = just_whitish.*not(just_blue);
% remove red from white
just_whitish = just_whitish.*not(just_red);
old_whitish = just_whitish; % used later task2func
% too white shouldn't be red or blue
just_red = just_red.*not(pre_whitish);
just_blue = just_blue.*not(pre_whitish);
% if it is too red shouldn't be blue, and also the countrary is true
just_blue = just_blue.*not(just_red);
just_red = just_red.*not(just_blue);
% cut on edges of the original image to get more components
just_blue = just_blue.*not(edges);
just_whitish = just_whitish.*not(edges);
```

With the goal of **separating regions** for each object in each binary image, we subtract the masks among them in this way. This method is conditioned from the color thresholding configuration.

Pre-processing

- Dealing with **very small signs** we needed to use no pre-processing on the original image and a weak pre-processing on the binarized ones
- The pre-processing here is then mainly oriented to **improve performances** of `regionprops` removing very small regions/noise
- For `just_red`, `just_blue` and `just_whitish` are used different hit-or-miss patterns and then used a logical `or` on the results
- The just_yellow not containing small interesting elements is just being "opened" with a diamond structuring element

```
imopen(just_yellow,strel('diamond',7));
```

Code for pre-processing

```
just_yellow = imopen(just_yellow,strel('diamond',7));

hm_i={[1 0 0; 0 1 0; 0 0 1]; [0 1 0; 0 1 0; 0 1 0];
    [0 0 1; 0 1 0; 1 0 0]; [1 0 0; 0 1 0; 0 1 0];
    [0 0 1; 0 1 0; 0 1 0]; [0 1 0; 0 1 0; 1 0 0];
    [0 1 0; 0 1 0; 0 0 1]; [0 0 0; 1 1 1; 0 0 0];
    [0 0 0; 1 1 0; 0 0 1]; [0 0 1; 1 1 0; 0 0 0];
    [0 0 0; 0 1 1; 1 0 0]; [1 0 0; 0 1 1; 0 0 0]};
for i=1:20
    fjust_red = bwhitmiss(just_red,hm_i{1});
    fjust_blue = bwhitmiss(just_blue,hm_i{1});
    fjust_whitish = bwhitmiss(just_whitish,hm_i{1});
    for j=2:size(hm_i,1)
        fjust_red = fjust_red|bwhitmiss(just_red,hm_i{j});
        fjust_blue = fjust_blue|bwhitmiss(just_blue,hm_i{j});
        fjust_whitish = ...
            fjust_whitish|bwhitmiss(just_whitish,hm_i{j});
    end
    just_red = fjust_red;
    just_blue = fjust_blue;
    just_whitish = fjust_whitish;
end
```

Some results

Region analysis

- For each binary image we obtain a label matrix of the connected components
- We are interested in a very specific kind of region in the binary images in terms of `BoundaryBox` size
- If a region in the label matrix complies then is added to the hypothesis for the correspondent binary image

Code for just_blue hypothesis

An example for just_blue:

```
1  %get outlines of each BLUE object
2  [¬,L,N] = bwboundaries(just_blue,4);
3  %get stats
4  stats =  regionprops(L,'BoundingBox');
5  BBox = cat(1,stats.BoundingBox);
6
7  hypBlue = []; %hypothesisBlue
8
9  for i=1:N
10     x = BBox(i,1); y = BBox(i,2);
11     width = BBox(i,3); height = BBox(i,4);
12     if abs(width-height)<abs(mean([width height])*0.5) && width ...
           < 150 && width > 10 && height >10 && height < 150
13         hypBlue = [hypBlue; [y y+height x x+width]];
14     end
15 end
```

hypothesisRed and hypothesisRedExt

```
1  if abs(width-2*height)<width*0.4 && width < 150 && width > 10 ...
       && height > 5 && height < 70
2      if abs(Centroids(i,1)-center(1))<width*0.1
3          if Centroids(i,2)-center(2)>0
4              hypRedExt = [hypRedExt; [y-height*1.5 y+height x-2 ...
                   x+width+2]];
5          else
6              hypRedExt = [hypRedExt; [y y+height*2.5 x-2 x+width+2]];
7          end
8      end
9  end
```

- hypothesisRedExt is obtained looking specifically for something that can be "half of a proibitory signal"
- Centroids are used to extend the ROI for the hypothesis in the right direction and provide eventually the "reduced classifier" with the complete sign

`hypothesisWhite` and `hypothesisYellow`

- White and yellow regions often represent only the central part of a sign
- The regions of interest for `hypothesisWhite` and `hypothesisYellow` are obtained enlarging the bounding boxes uniformly in all directions in order to provide the reduced classifier with the entire sign

Combine all the hypothesis

Hypothesis are sorted in order of "accuracy", the methods that are more likely to catch a lot of noise due to blind enlargement of the ROI are last.

```
% order of hypothesis matters
hypothesis = [hypothesisRedExt; ...
    hypothesisBlue; hypothesisRed; ...
    hypothesisYellow; hypothesisWhite];
```

Reduced classifier

- The hypothesis from the region analysis are used to cut a window in the **old binary images**, those are passed to a reduced version of the classifier used in task 2, modified to work in the absence of a sign and to return a **verdict** and a **more precise ROI**
- The noise removed with mask subtractions and pre-processing is re-introduced to give more details to the classifier

Call to classifier

```
for i = 1 : size(hypothesis,1)
    hy=floor(hypothesis(i,:));
    windowRed = old_red(max(1,hy(1)):min(hy(2),ors(1)), ...
        max(1,hy(3)):min(ors(2),hy(4)),:);
    windowBlue = just_blue(max(1,hy(1)):min(hy(2),ors(1)), ...
        max(1,hy(3)):min(ors(2),hy(4)),:);
    windowWhitish = old_whitish(max(1,hy(1)):min(hy(2),ors(1)), ...
        max(1,hy(3)):min(ors(2),hy(4)),:);
    windowYellow = just_yellow(max(1,hy(1)):min(hy(2),ors(1)), ...
        max(1,hy(3)):min(ors(2),hy(4)),:);
    [Verdict, newROI] = ...
        task2func(windowRed,windowBlue,windowWhitish,windowYellow);
    if Verdict == 1
        if numel(newROI)≠0
            ROI_conf = [hy(1) hy(1) hy(3) hy(3)]+newROI;
        else
            ROI_conf = hy;
        end
        [...]
    end
end
```

Where `ors` is the size of the original image

Positive verdict

```
% Prevent overlaps
found=0;
for sfi=1:size(signs_found,1)
    center_sfi = [mean(signs_found(sfi,3:4)) ...
        mean(signs_found(sfi,1:2))];
    center_roi = [mean(ROI_conf(3:4)) mean(ROI_conf(1:2))];
    if(norm(center_sfi-center_roi)<20)
        found=1;
    end
end

if found==0
    signs_found = [signs_found; hyp];
end
```

The reduced classifier

The steps performed by the classifier are in order:

- Circles detection
- Lines detection
- Builds masks with shapes found in signs to be applied to the input binarized images, masks are adjusted depending on the features found in the previous 2 steps
- Compute the overlap areas among some input binary images and the shape masks to determine "scores"
- Cascade of `ifs` to determine if it is a sign, at this stage we don't care about the class

Circles detection

Red and Blue circles are detected toghether with circular blanks inside red zones, the radii are adjusted to the window size

```
1  %dinamically adjust radii filter
2  Rmax = ceil(max([max(window_size)/2 6]))+2;
3  Rmin = ceil(max([min([Rmax/2 min(window_size)/2]) 8]));
4
5  % Find all the blue bright circles in the image
6  [centersBlueBright, radiiBlueBright] = ...
       imfindcircles(blueMask,[Rmin ...
       Rmax],'ObjectPolarity','bright','Sensitivity',0.90);
7  % Find all the blue bright circles in the image
8  [centersRedBright, radiiRedBright] = ...
       imfindcircles(redMask,[Rmin ...
       Rmax],'ObjectPolarity','bright','Sensitivity',0.90);
9  % Find all the blue dark circles in the image
10 [centersRedDark, radiiRedDark] = imfindcircles(redMask, [Rmin ...
       Rmax],'ObjectPolarity','dark','Sensitivity',0.90);
```

Line detection on redMask

Lines are detected only at specific theta angles and then filtered only for specific rho distances. We are interested just in the red lines that can be the edges of a triangular sign.

```
rng_theta = {-15:-0.25:-55; -15:-0.25:-55; 15:0.25:55; 15:0.25:55};
rng_rho = [-0.30 0.40;0.1 0.7;0.6 1.2;0.2 0.7]*mean(window_size);
red_lines = [];
redl_sides = [0 0 0 0];
for i = 1 : size(ranges_theta,1)
    [H,T,R] = hough(redMask,'Theta',ranges_theta{i});
    P  = houghpeaks(H,2,'threshold',ceil(0.2*max(H(:))));
    lines_temp = houghlines(redMask, T, R, P, 'FillGap', 1, ...
        'MinLength', window_size(1)/2);
    if size(lines_temp)≠0
        for j=1:size(lines_temp)
            if lines_temp(j).rho > ranges_rho(i,1) && ...
                lines_temp(j).rho < ranges_rho(i,2)
                red_lines = [red_lines; lines_temp(j)'];
                redl_sides(i) = redl_sides(i) + 1;
            end
        end
    end
end
```

Some of the masks used

```
1  %% MASKS
2  if size(centersRedBright,1)==1
3      circ_mask1 = ...
           circularMask(centersRedBright,radiiRedBright,window_size);
4      area_mask1 = pi*(radiiRedBright)^2;
5  elseif size(centersBlueBright,1)==1
6      circ_mask1 = ...
           circularMask(centersBlueBright,radiiBlueBright,window_size);
7      area_mask1 = pi*(radiiBlueBright)^2;
8  else
9      circ_mask1 = ...
           circularMask(window_size/2,mean(window_size)/2,window_size);
10     area_mask1 = pi*(mean(window_size)/2)^2;
11 end
12
13 tria_mask3 = roipoly(redMask,[window_size(2)/2 window_size(2)/4 ...
       3*window_size(2)/4],[window_size(1)/4 3*window_size(1)/4 ...
       3*window_size(1)/4]);
14 tria_mask2 = roipoly(redMask,[window_size(2)/2 1 ...
       window_size(2)],[1 window_size(1) window_size(1)])-tria_mask3;
15 %[...]
```

An example of score calculation

```
%% SCORING
score_blue1 = sum(sum(circ_mask1.*blueMask))/area_mask1;
score_red1 = sum(sum(circ_mask1.*redMask))/area_mask1;
score_white1 = sum(sum(circ_mask1.*whitishMask))/area_mask1;

score_red2 = sum(sum(tria_mask2.*redMask))/sum(sum(tria_mask2));
score_red4 = sum(sum(tria_mask4.*redMask))/sum(sum(tria_mask4));
%[...]
```

Basically computation of the overlapping area for the shape mask and color mask divided by the area of the shape mask.

An example from the "decision part"

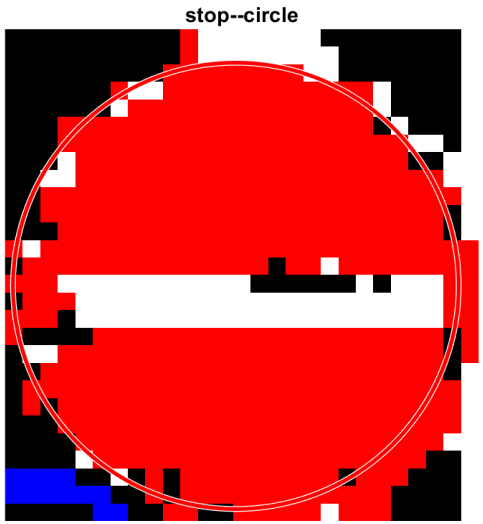
```
1  %[...]
2  %% DETECT PROHIBITORY
3   if score_red6 > 0.6 && score_red7 < 0.5 && score_yellow7 < 0.15
4       if size(centersRedBright,1)==1 && size(centersRedDark,1)==1
5           if score_red6 > 0.8
6               result = 'prohibitory2circles';
7               newROI = [centersRedBright(2)-radiiRedBright ...
                      centersRedBright(2)+radiiRedBright ...
                      centersRedBright(1)-radiiRedBright ...
                      centersRedBright(1)+radiiRedBright];
8           end
9       elseif size(centersRedBright,1)==1 && score_red6 > 0.6 && ...
            score_red7 < 0.6 && score_blue1 < 0.1 && score_white1 > ...
            0.35
10          result = 'prohibitory1circle';
11          newROI = [centersRedBright(2)-radiiRedBright ...
                    centersRedBright(2)+radiiRedBright ...
                    centersRedBright(1)-radiiRedBright ...
                    centersRedBright(1)+radiiRedBright];
12      end
13  end
14  %[...]
```

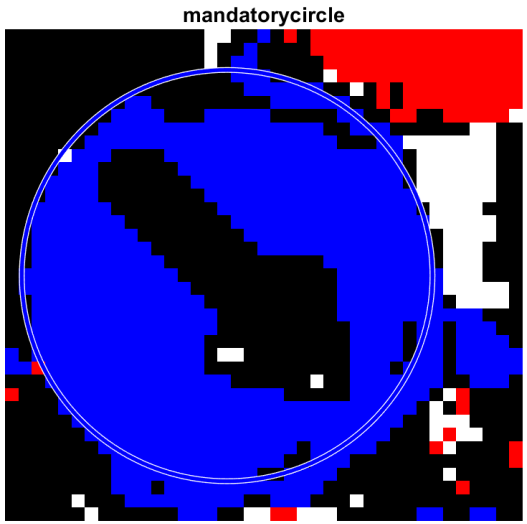
A quick example - original

After masks, subtraction, preprocessing

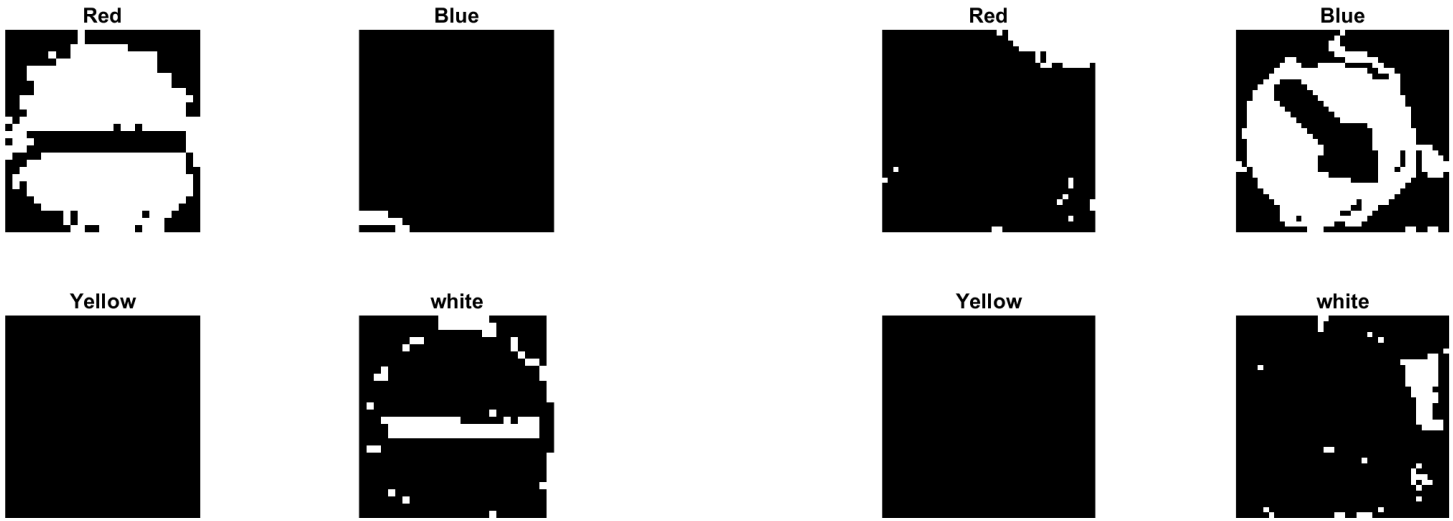
As seen from the classifier

As seen from the classifier





As seen from the classifier



Final result

The current version of the script has a score of

- `PRECISION 9.193548e-01`
- `RECALL 8.142857e-01`

on the test set provided

Those values can be easily improved with tuning and revision, but are good keeping into account the limited amount of time available.

Possible improvements

These scripts have been tested and putted together in a relatively short amount of time, some more debugging and some reorganization would already improve the results

- Implement a decision tree in the classifier, applies also to task 2
- Dynamical masks also for triangles
- ...

Task 2: Experiments and challenges

Goal: Classify road signs by their type ('mandatory', 'prohibitory', 'danger', 'other') using the ROI of a previous detection

- The identification is based on the scores obtained from multiplying color with shape masks and the number of lines and/or circles found
- The Red, Blue, White and Yellow masks from Detection were also used for Classification

Procedure overview

- Pre-processing mask computation
- Circles Lines
- Masking
- Scoring and Classifying

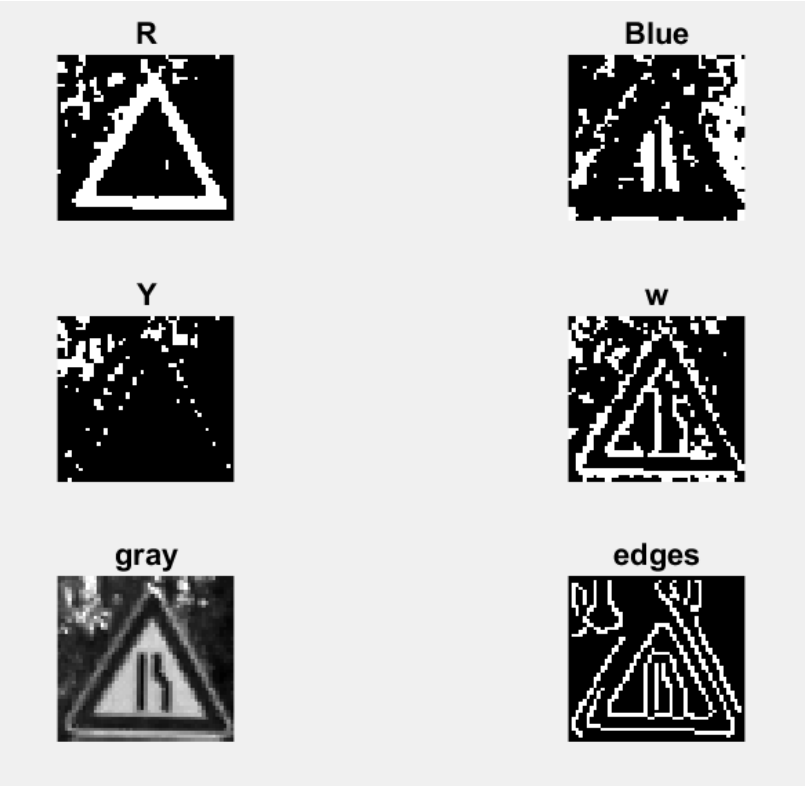
Pre-processing and colour mask computation

- Contrast adjustment and colour thresholding mask creation as follows (similar to task1)

```
redMask = maskRed(contrast);
blueMask = maskBlue(contrast);
yellowMask = maskYellow(contrast);
whitishMask = maskWhitish(contrast);
gray = rgb2gray(contrast);
edges = edge(gray,'canny');
```

Pre-processing and colour mask computation

j



Circles and lines

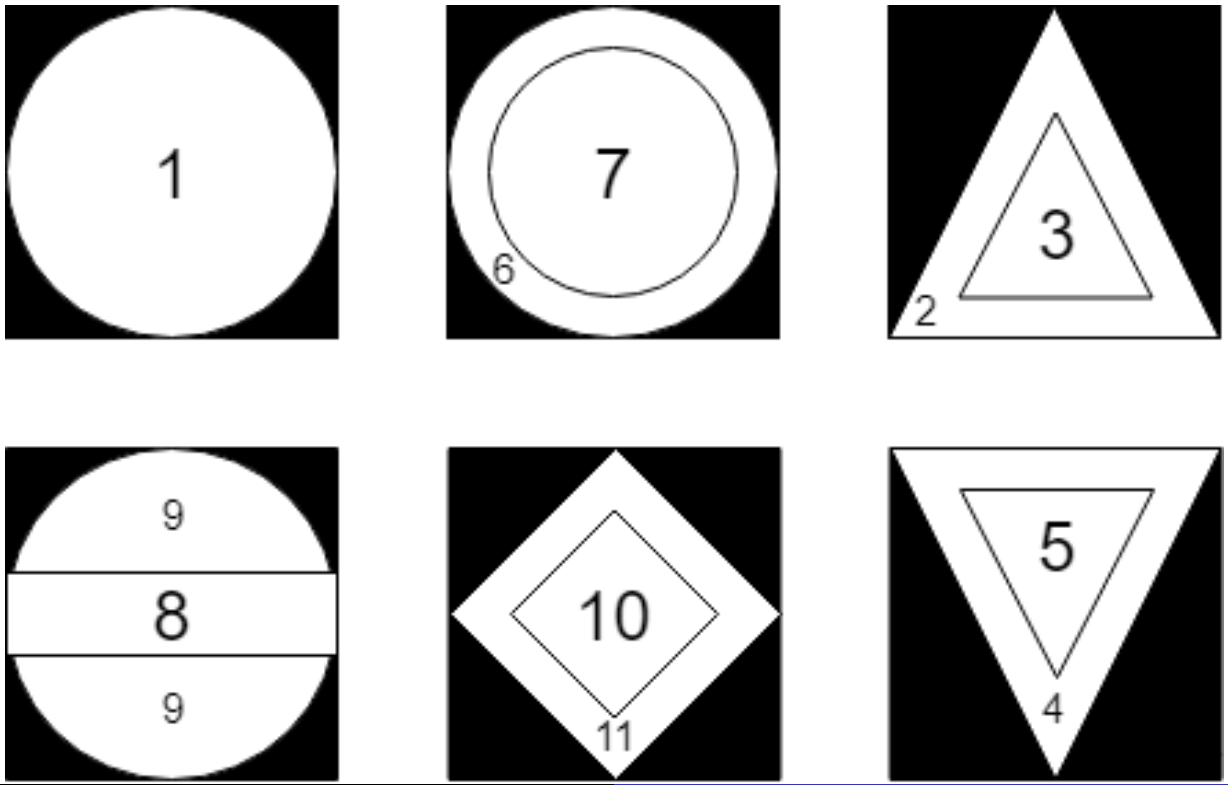
- Find circles with dynamic radius range in gray, blue red masks - prohibitory, mandatory, forbidden
- Find lines in edges and red masks - stops, priority, danger

Shape masks

- 11 different masks to suit our needs (triangle, circle, diamond, misc.)
- Circular ones are dynamically based on the size of the circle we found

Scoring and Classifying

Hadamard product of colour masks from original picture and ideal shape masks



Results

With this approach, the results were very good. The current version can recognize 99

Task 3: Experiments and challenges

Goal: Given a single-sign image of a specific class traffic sign, recognize the traffic sign.

Challenges:

- It has to be done a proper mask for each specific class of traffic signs.
- Blurred images are difficult to pre-processing.
- Choose the best features for each type of sign of a single class of traffic sign.
- Bad masks led to a bad recognition of the sign.

Pre Processing

First of all, we resized the images (400x400) to work on same size.
Afterwards, we defined the PreProcessing function that performs the first part to get the final mask of the signal.
We should define a different Pre-processing function for each class of signal:

- Danger Pre-Processing
- Mandatory Pre-processing
- Prohibitory Pre-processing

Pre Processing

For example, for danger signs we followed these steps:

- Sharpen the image and deblurr it.
- Transform it in gray scale.
- Define a triangle mask, to remove not interesting objects.
- Equalize the image with CLAHE adaptive equalization.
- Binarize it and perform erosion on the complement of the mask to divide regions.

Masks

After got a processed image, we performed the Regionprops function to be able to define the central region of the sign and remove the small objects around (noises). We created 2 functions to do that:

- AreaConstraint(binmask,smallestArea);
- DefineCentralRegion(binmask);

They both receive in input the mask that comes out from the Pre-processing function. AreaConstraint also receive the size of limit area allowed to exist in the image.

Sign characterization: Introduction

The approach that we chose is to find, for each type of sign of each class, its main features.
To do that we define a TrainingData function.
This function is able to define the main features of a sign and it gives them back to the main program so we can compare them with the currently processed sign's features.

TrainigData function

The steps of this function are:

- mask the sign (with Pre-Processing and Regionprops functions)
- use Regionprops function again to define features of the mask and also on the ConvexHull mask got from it.
- find max,min and average values of every feature.
- collect all the data in an array of structs to give it back to the main program.

TrainingData function

The features that we collect are:

- MajorAxisLength
- MinorAxisLength
- Area
- Perimeter
- Eccentricity
- Centroid

Array of features from TrainingData

Here below the arrays that contain the features of the mask.

Fields	N_objects	max_Perir	min_Perir	max_area	min_area	max_Eccent	min_Eccent	max_centr	max_centr	min_centr	min_centr	max_maxAx	min_maxAx	max_minAx	min_minAx
1	3	299.3840	134.7460	2155	1401	0.9482	0.2046	213.4527	313.9367	192.4997	175.2066	116.3503	45.5260	50.3215	36.9667
2	2	343.4260	270.5470	2853	1382	0.9965	0.9837	247.5979	279.9846	181.2915	252.2699	177.7573	144.7206	28.2486	12.6186
3	1	346.2470	315.0200	3844	2976	0.9141	0.9102	235.1358	272.4378	215.8962	261.0444	144.6197	129.4416	58.6406	53.6126
4	1	392.3040	326.7630	4848	2651	0.9311	0.9119	187.4865	274.3331	178.5736	243.4142	173.7061	135.1919	63.5872	53.5113
5	1	532.6430	414.2060	9586	6621	0.8477	0.7082	208.9726	258.7436	184.5715	236.7779	149.9930	118.4730	95.2224	74.7566
6	1	497.6140	400.0190	7512	3476	0.8924	0.8604	229.1604	272.0625	207.2908	243.9790	168.5226	138.3087	82.1565	62.4183
7	1	863.4920	691.5430	5507	3729	0.5680	0.5237	211.0799	270.7566	203.6886	261.5701	132.6225	115.1662	110.9790	98.1092
8	2	326.9920	103.6830	4078	867	0.9848	0.2992	214.2890	317.2032	189.3329	184.9060	160.5968	35.7355	35.8631	25.9985

Fields	C_max_are	C_min_are	C_max_Peri	C_min_Peri	C_max_Ecce	C_min_Ecce	C_max_cent	C_max_cent	C_min_cent	C_min_cent	C_max_max	C_min_max	C_max_min	C_min_min
1	7939	6377	402.5060	376.0080	0.9700	0.9609	213.1523	253.9671	196.9214	237.1353	193.5443	183.8667	53.5769	44.9951
2	10684	8384	416.9440	364.4130	0.8748	0.8337	225.6044	280.2487	209.0131	257.1621	169.1626	146.3587	84.5476	75.4699
3	5954	4922	313.8790	281.7690	0.8681	0.8620	230.5032	276.1747	213.0004	266.6904	127.1898	114.0108	63.1259	57.7861
4	7683	4974	358.1250	297.4230	0.8970	0.8646	191.8547	277.9787	183.2141	249.0585	150.3534	120.3491	67.9746	57.7003
5	13776	9133	439.4150	360.7260	0.6914	0.6052	209.7492	263.7813	183.7762	239.6990	152.5243	123.0831	117.0529	97.4222
6	11123	6439	408.5760	330.4300	0.8715	0.8210	229.0964	272.3676	207.6021	242.8249	159.0536	131.1131	90.8136	64.2865
7	10909	8390	381.7620	332.2480	0.5400	0.4829	212.5111	267.0934	203.1564	261.1313	128.9323	111.8855	109.0983	96.0833
8	6952	5726	432.4760	408.9480	0.9878	0.9783	211.9484	239.4770	194.5267	219.7919	221.6426	206.6260	42.8548	34.2769

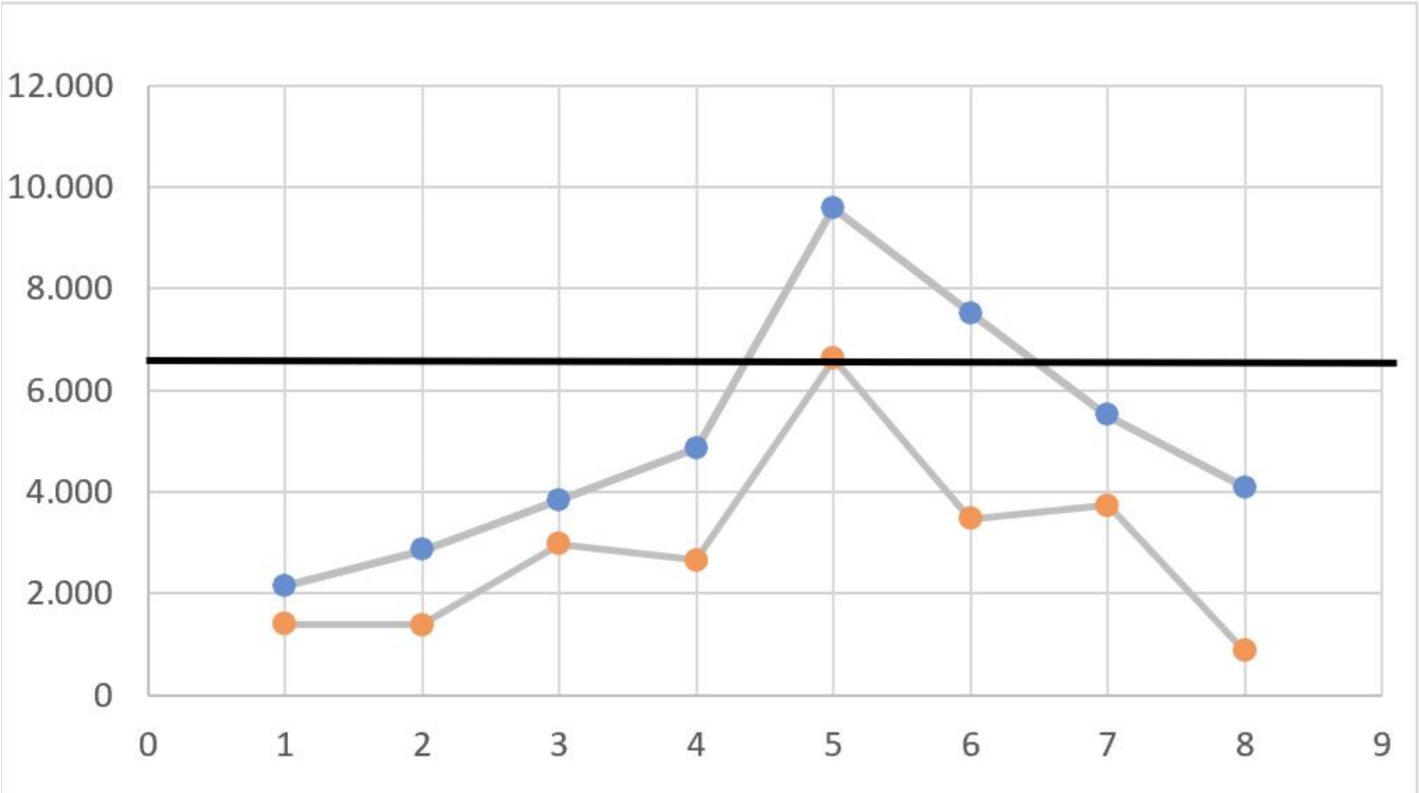
Starting classification

It is done with an if-else structure after divide the sign into the number of regions in the mask. For example, to get crossroad sign:

```
1  if confront.Area > datas(crossroad).min_area
2      if confront.Area > (datas(scurve).max_area + 200)
3          figure;imshow(original1);title('cross road');
4      else
5          if confront.Area < (datas(crossroad).min_area - 300)
6              figure;imshow(original1);title('S curve');
7          else
8              if confront.Centroid(1,1) > datas(crossroad).max_centr_x
9                  figure;imshow(original1);title('S curve');
10             else
11                 if  confront_Hull.Eccentricity > ...
                       data_Conv(crossroad).C_max_Eccentricity
12                     figure;imshow(original1);title('S curve');
13                 else
14                     figure;imshow(original1);title('cross road');
15                 end
16             end
17         end
18     end
19 end
```

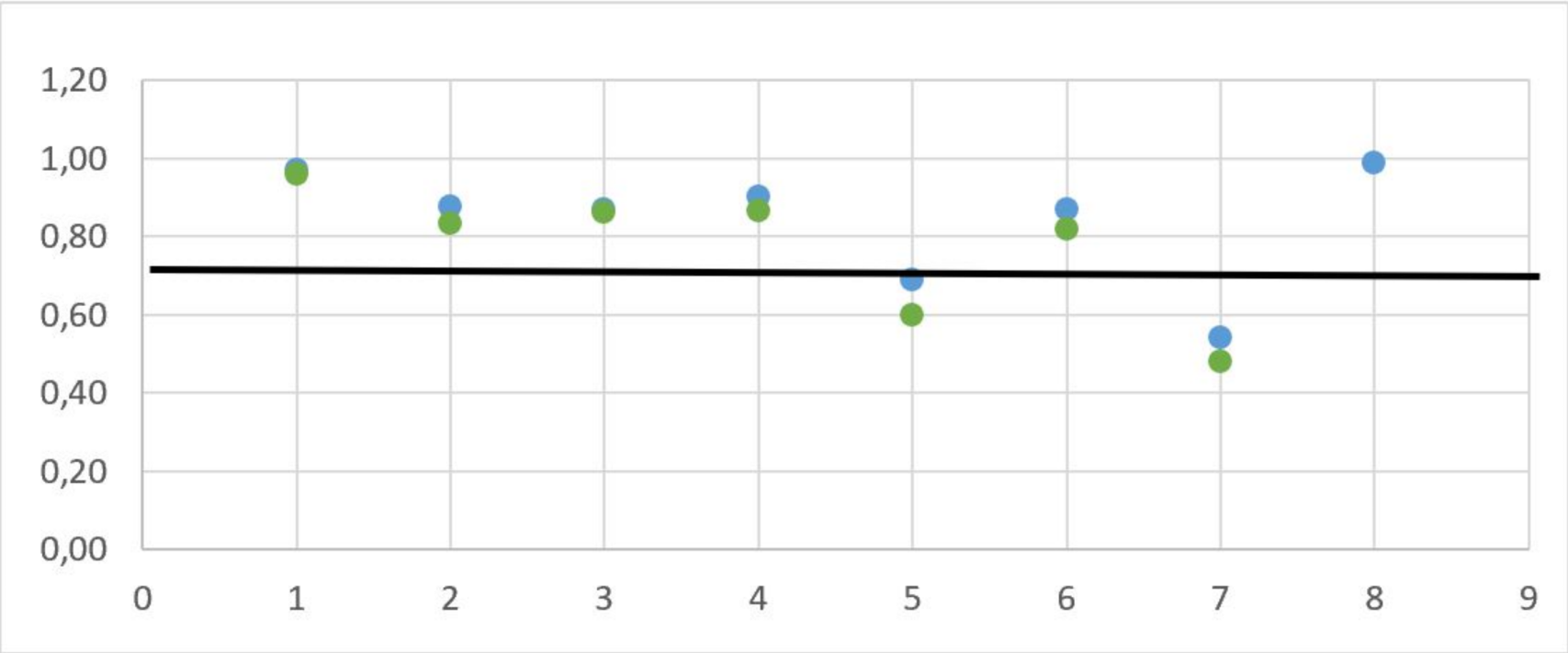
Starting classification

Below: Max and min Area of each danger signs. Crossroad sign has the bigger area. S-curve is in the lower range of the Crossroad sign. We have to distinguish them.



Starting classification

Max and min Eccentricity of each danger sign. Crossroad sign has lower Eccentricity with respect to S-curve sign.



Results

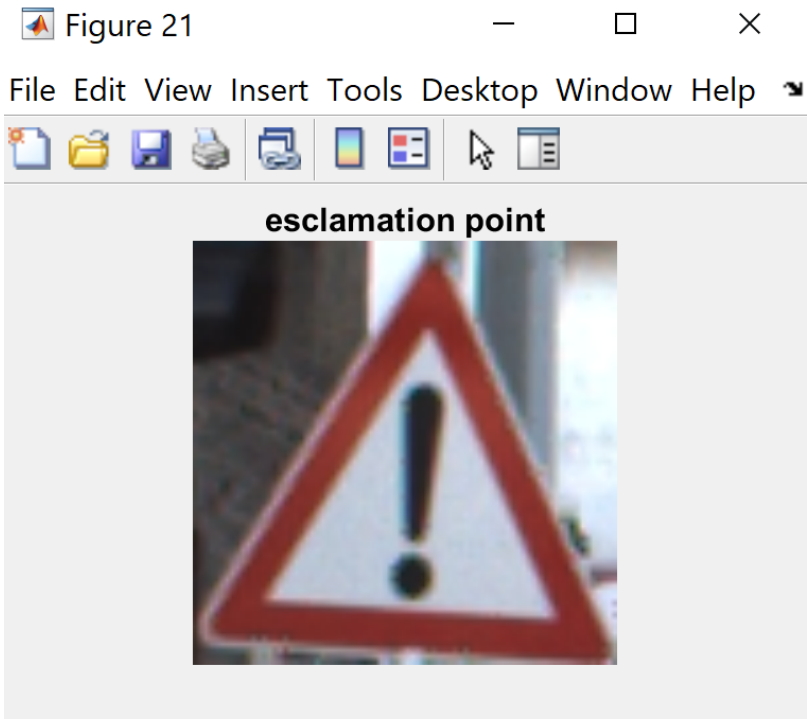


Figure 21
File Edit View Insert Tools Desktop Window Help
esclamation point

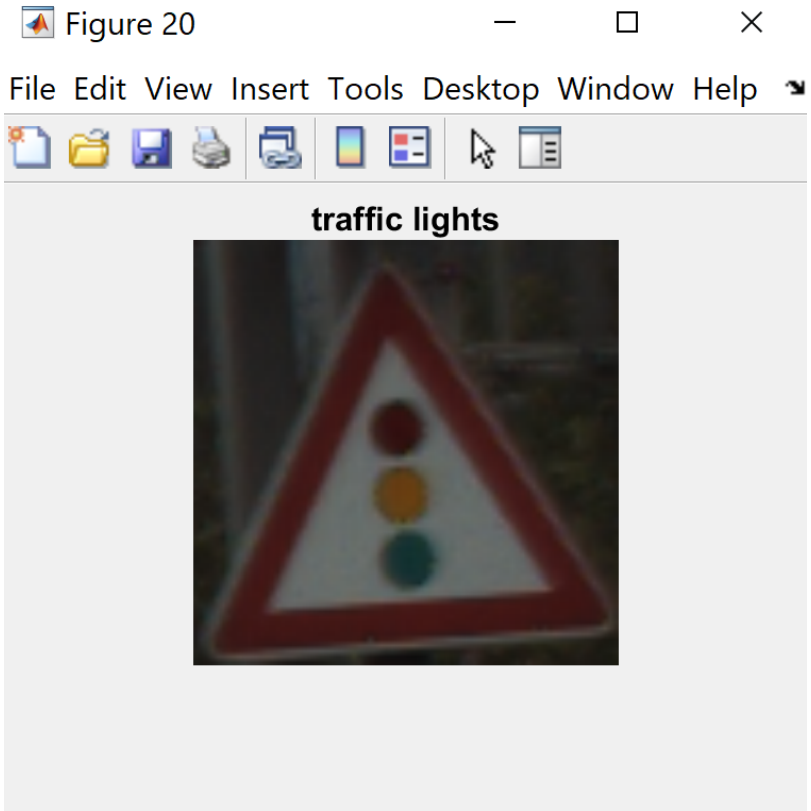


Figure 20
File Edit View Insert Tools Desktop Window Help
traffic lights

Results

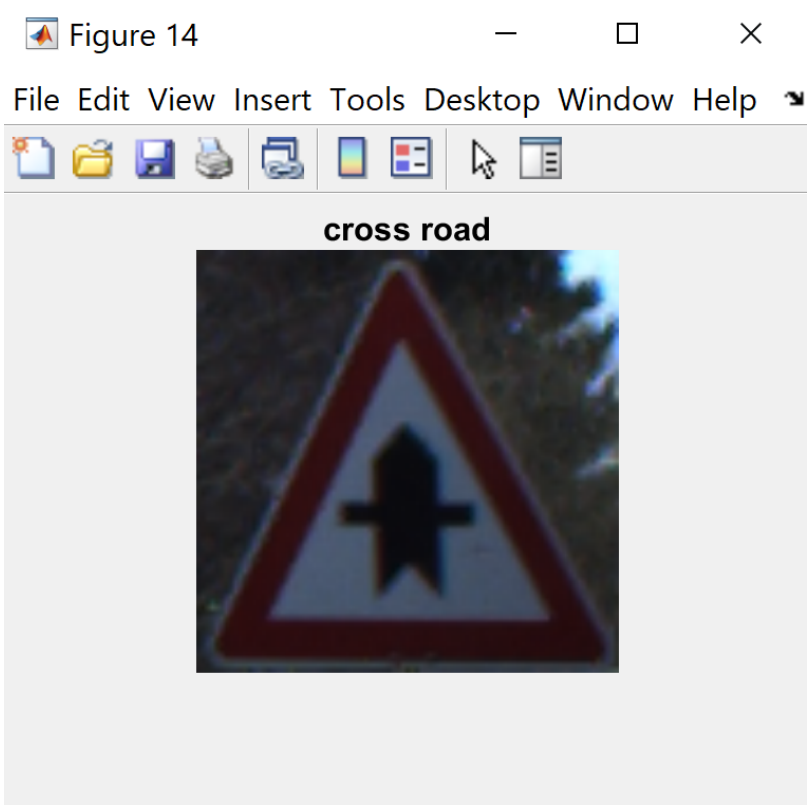


Figure 14
File Edit View Insert Tools Desktop Window Help
cross road

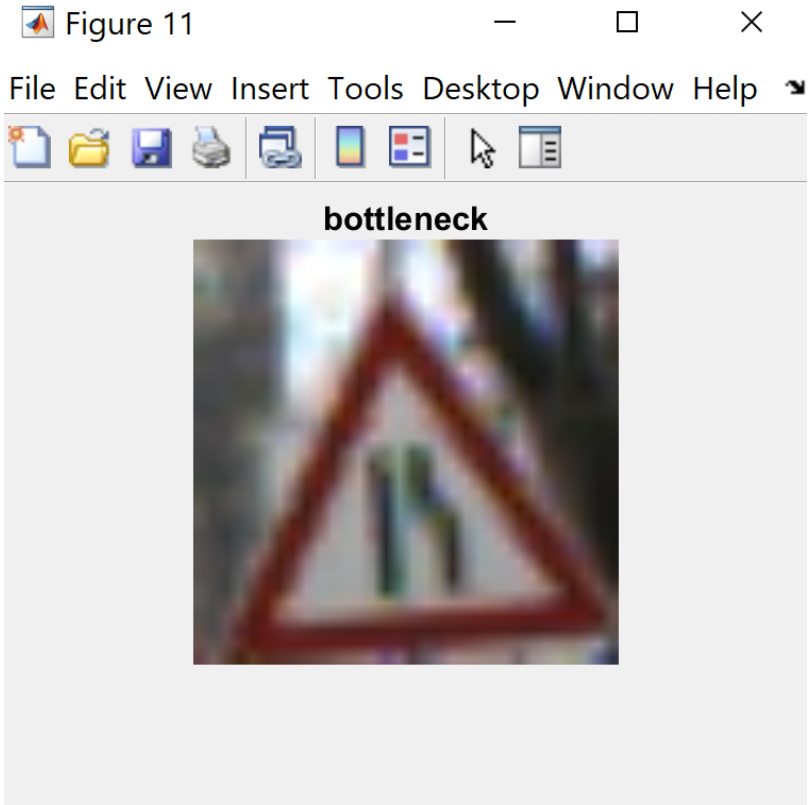


Figure 11
File Edit View Insert Tools Desktop Window Help
bottleneck

Consideration

The algorithm works for danger signs.
Due to lack of time we didn't tried it with mandatory and prohibitory signs but the procedure is the same.
Only the pre-processing process change.

Possible improvements

It has to be done the same for every type of sign. Anyway, the algorithm can be improved, for example by:

- Performs a better pre-processing.
- Introduce more features to distinguish sign.
- Taking into account also colors to distinguish sign.

Of course, if an image is to blurred, it's very difficult to mask the sign.

Conclusions

We are satisfied by the results that we got from our work.
As we said in every sections, the algorithm can be optimized in different ways.
We spent almost all of our time looking to improve our initial solution by trying different methods to get it better.
We also had to delete all that we had done because we found a better solution.
The project-work increased our knowledge about image analysis and it could be consider a nice starting point to study deeply this topic.

Contributions

- Task 1:
  - Planning: L.Rinelli, T.Abreu
  - Coding: L.Rinelli
  - Slides: L.Rinelli
  - Report: L.Rinelli, T.Abreu
- Task 2:
  - Planning: L.Rinelli, R.Alves
  - Coding: L.Rinelli
  - Slides and report: T.Abreu, R.Alves, L.Rinelli
- Task 3:
  - Planning: M.Dolci, L.Rinelli
  - Coding: M.Dolci
  - Slides and report: M.Dolci

Thank you for your attention![2]

[2]From today without palm oil