Computer Vision 2021 - Homework (LAB 3): *Street features detection*

Michele Guadagnini - ID 1230663

July 9, 2021

Abstract

The goal of this homework is to detect and highlight the **street lanes** and the **round street signs** from some pictures. The program exploits the *Canny* edge detection algorithm and the *Hough* transform to complete the task. The detected features will be highlighted by coloring the corresponding pixels.
The code has been developed in *C++* language, making use of the *OpenCV* library. Almost all the parameters of the applied algorithms can be set making use of trackbars; this allows to easily select the best parameters.

1 Implementation description

The solution of the task has been developed by creating a library containing the functions to be called in the main program. The most important ones are:

- *DetectEdges*: it uses the Canny algorithm to detect the edges to be passed to the Hough transform. The function convert the input image to gray-scale and creates the two trackbars needed to tune the two threshold of the algorithm.
- *RetrieveLines*: it applies the Hough transform to the edges computed above in order to detect the street lanes. It creates 5 trackbars for the parameters. When using it, the parameters has been tuned to detected the right-most street lane, although it was not always possible.
- *RetrieveCircles*: it is used to detect the round signs in the images. Before passing the image to the *OpenCV* function **cv::HoughCircles**, a gaussian filter is applied in order to reduce the noise; the *sigma* of the filter must be passed as a parameter to the command line. Then, the function converts the image to gray-scale and creates the trackbars.
- *SaveResult*: it overlaps the detected regions with the colors defined at the top of the library implementation file *street_features_utils.cpp*: circles will be filled with green, lines are drawn with red and the region between lines with purple.

The library contains some other functions used to apply the gaussian filter and to compute the points representing the polygon of the region between detected lines.
Also a callback function for each of the first three functions above have defined, that are used when the parameters values are changed using the corresponding trackbar. To pass

the image and some parameters to the callbacks, a simple class has been defined (see *street_features_utils.h*).

The program requires 4 arguments to be passed on command line:

- the name of the input image file;
- the name of the output file;
- a string that contains the type of desired output. The possible options are "*region*", which overlaps a colored polygon to the region between the detected lines; "*lines*", which draws only the lines; "*both*", which plots both lines and region.
- the σ of the gaussian filter to be applied before circles detection. If $\sigma < 2$, filtering is skipped.

In case of wrong number of arguments, the program prints a help message with instructions.

1.1 Compilation and usage

The program has been compiled with the following command:

```
g++ -o street_features street_features_main.cpp street_features_utils.cpp `pkg-
  config --cflags --libs opencv`
```

Below it is reported a usage explanation of the executable:

```
./street_features [input_name] [output_name] [plot_option] [gaussian_sigma]
```

2 Results

The provided images has been analyzed with the program. The optimal parameters used in the analysis are reported in Table 1, Table 2 and Table 3.

Below instead are reported some comparisons between original and successfully processed images (Figure 1 and Figure 2). The target for the lines detection was the right-most street lane, although sometimes filter out the spurious lines has not been possible without additional conditions with respect to the parameters on trackbars (see Figure 3).

Other results are included the *.zip* archive.

Canny edge detection		
image	low threshold	high threshold
road1	436	917
road2	495	862
road3	192	302
road4	490	730
road5	628	885
road6	530	768
road7	148	355
road8	271	816

Table 1

Hough transform lines detection					
image	ρ res.	θ res.	Counts	min θ	max θ
road1	1	2	170	2	115
road2	1	1	105	0	180
road3	1	2	280	0	180
road4	1	2	129	0	180
road5	2	2	92	109	123
road6	2	2	47	136	180
road7	1	2	83	0	180
road8	1	2	98	0	180

Table 2

Hough transform circles detection						
image	min distance	canny high	Counts	min radius	max radius	gaus. σ
road2	10	100	23	1	500	4
road3	200	96	24	1	77	3
road4	5	115	16	1	500	4
road7	20	70	15	1	500	4
road8	20	32	100	1	280	4

Table 3

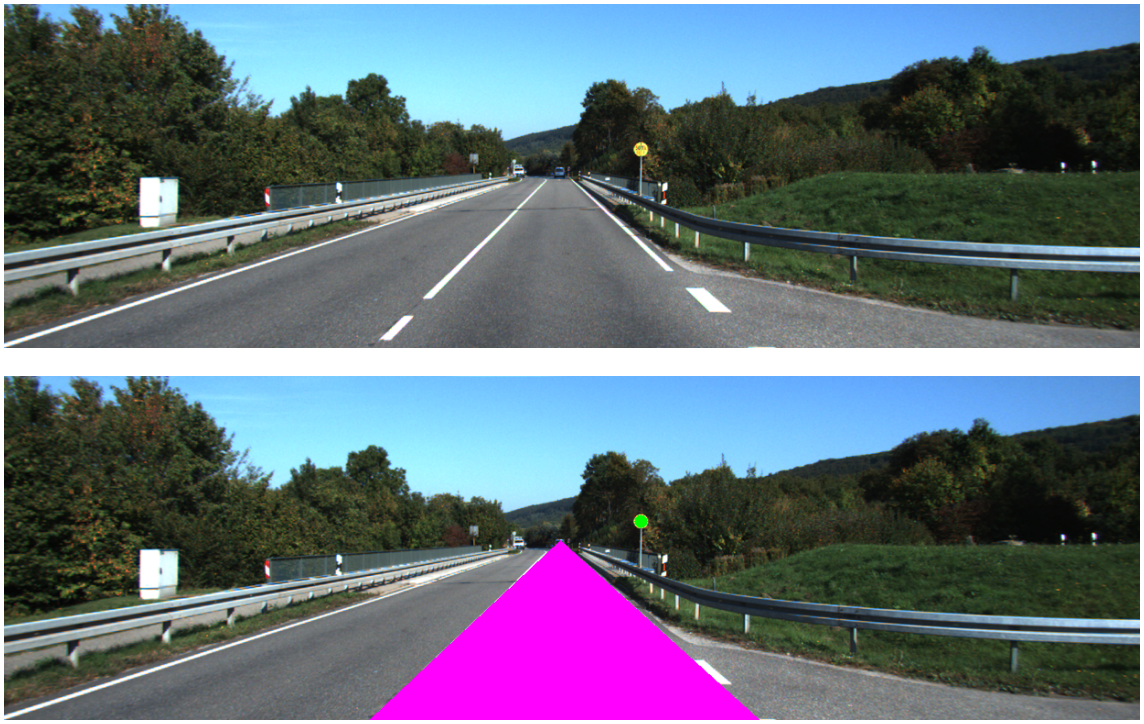

Figure 1

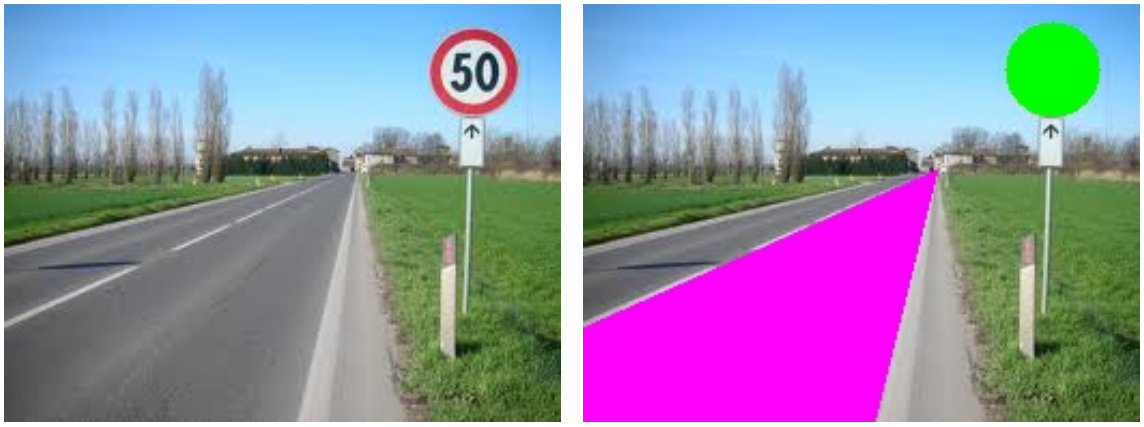

Figure 2

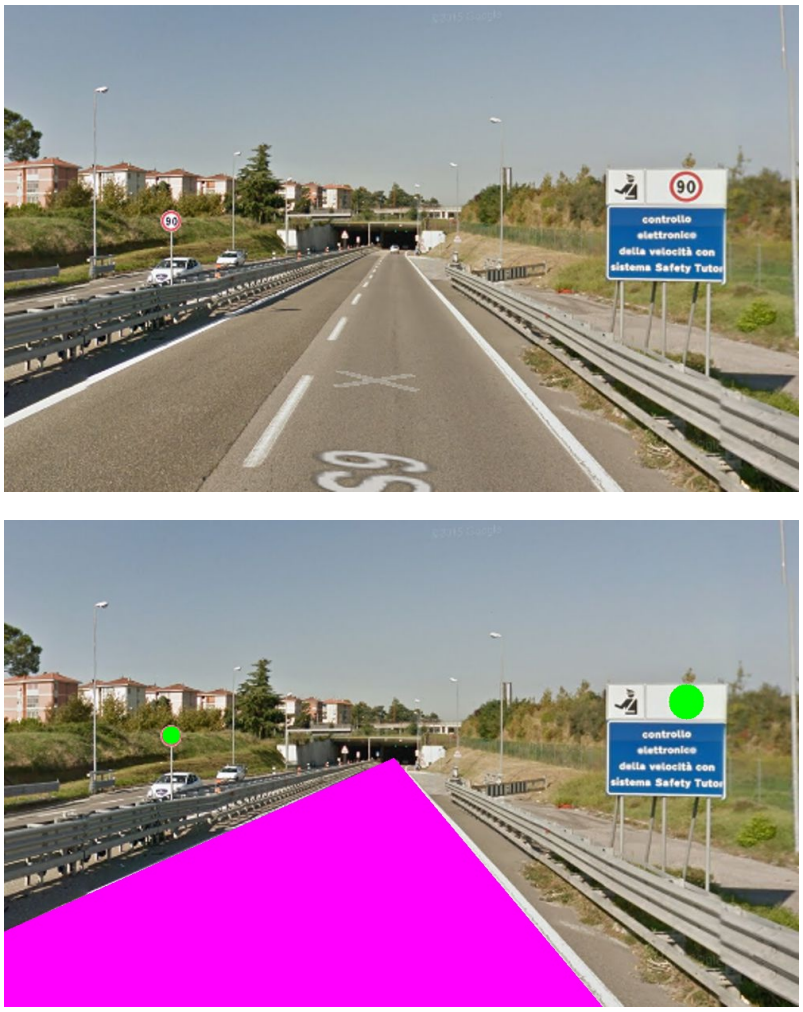

Figure 3