Lab 3: Edge, lines and circles detection

Nicole Zattarin

Abstract

ciao belli

1. Setup and parameters tuning

At this link we provide a possible implementation of a program that, given an image, performs edge, lines and circles detection. The executable is structured in order to receive from command line a path to an image, a path to a file of parameters and three boolean variables, that allow the user to decide if he wants to tune the parameters by means of trackbars, i.e. :

```
./main <image path> <params file path> <tune canny> <tune HoughLines> <tune HoughCircles>
```

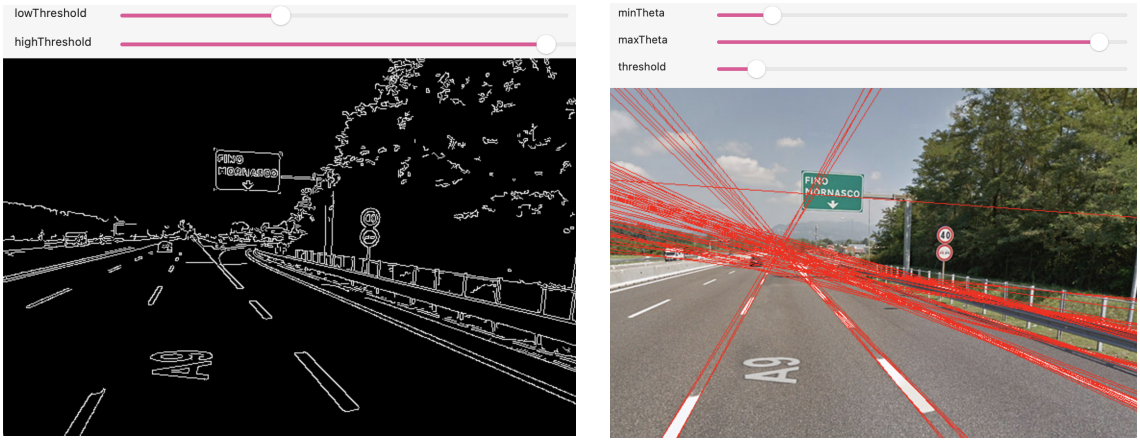
For instance, the following example launch the program in order to process the image `road2.png` getting initial parameters from the file `paramsRoad2.txt` and perform only the tuning of Canny parameters:

```
./main images/road2.png params/paramsRoad2.txt 1 0 0
```

Note that once a tuner flag is set to 1, the code runs up to that tuning phase with the parameters provided in the file, then once the tuning window is closed, the execution stops as well. The idea is that we can get a good result by tuning a detector at a time, i.e. tune a detector for each run of the code, moreover, everything is made pretty efficient by the usage of trackbars. Therefore, first we tune Canny, see an example of interface in Figure 1a, and we write down the best set of parameters in the `.txt` file. Then, in two successive runs, we do the same with HoughLines, see Figure 1b, and HoughCircles. Trackbars are useful to see in real time how the output of a detector is affected by parameters change, for instance in Figure 1b we can observe lines appearing and disappearing according to the angle range and to the threshold. At the end of this tuning process the chosen parameters are printed on the screen, so that we can register them manually in the `.txt` file, in such a way that running the algorithm on the same image will return the desired result and save the corresponding images in a specific folder. To summarize, the workflow should look like:

```
./main images/road2.png params/paramsRoad2.txt 1 0 0 // manually tune params, update params file
./main images/road2.png params/paramsRoad2.txt 0 1 0 // manually tune  params, update params file
./main images/road2.png params/paramsRoad2.txt 0 0 1 // manually tune params, update params file
./main images/road2.png params/paramsRoad2.txt 0 0 0 // get all final results!
```

Trackbars callbacks are collected in `trackbars.h`, while `utils.h` contains all the functions that can be useful to manipulate images, lines, circles and draw on the image itself. Finally, note that each of the image requires a specific tuning and a specific processing in order to get the final result, this is performed with functions provided in `images_processing.h`. Therefore, running the code on a generic image won't give back the same result as on the samples ones, unless one creates a specific function to deal with it.



(a) *Example of interface for Canny parameters tuning, i.e. lower and higher thresholds.*

(b) *Example of interface for HoughLines parameters tuning, i.e. threshold and angle range.*

Figure 1: *Example of interface for a run in which we ask to tune the Canny parameters (a) and the HoughLines (b). Note that parameters are set randomly in this example, just to show that the image changes according to values inserted through the trackbar.*

2. Results

We test edges, lines and circles detection of three images provided in the folder `images`, in particular `road2.png`, `road3.jpg` and `road4.jpg`. The initial part of the processing is the same for all the images and consists of the following steps:

1. Load image and parameters, show the original image;
2. Apply Canny to the greyscale image, tune the parameters if it's the case and save the edge map;
3. Use the output of Canny, i.e. the edge map, to individuate lines within the image by means of HoughLines, tune the parameters in order to individuate only the boundaries of the street or of the street lane. Moreover, once the parameters are fixed, we fill the polygon determined by the lower points of two intersecting lines and the intersection point;
4. Apply HoughCircles the greyscale image to the greyscale image to individuate circles, with particular attention to the most relevant parameters in order to detect only round street signs. Finally we fill them with the same color.

In particular parameters to process image `road2.png` are shown in Table 3, while in Figure 2 we provide three different steps of the image manipulation: original image in Figure 2a, edge map in Figure 2b and the final result in 2c. Note that in the final result we show only the relevant lines and we fill in between only up to the intersection point.

For what concerns image `road3.jpg`, parameters are shown in Table **??**, while images in Figure 3.

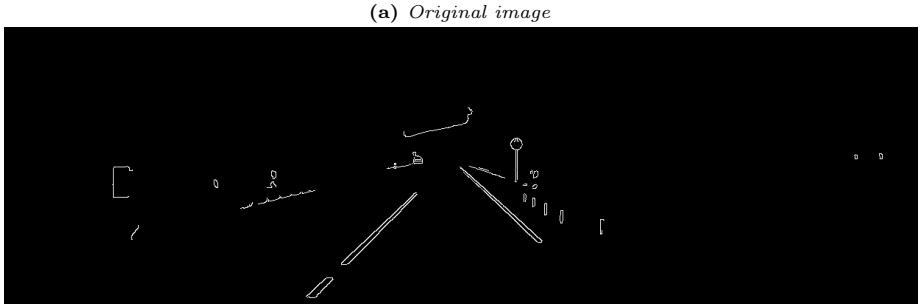
Canny	apertureSize	3
	threshold1	350
	threshold2	850

HoughLines	rho	1
	theta	0.05
	threshold	130
	min theta	0
	max theta	CV PI

HoughCircles	dp	1
	minDist	1
	param1	100
	param2	25
	minRadius	0
	maxRadius	10

Table 1: *Parameters set for road2.png image, for all the three algorithms.*

(a) *Original image*



(b) *Edge map after applying Canny to the greyscale image*

(c) *Final result: circle street signs are filled in green, while the area between the intersected lines is filled in magenta*

Figure 2: *Three different steps of the image manipulation: original image in Figure 2a, edge map in Figure 2b and the final result in 2c.*

Canny	apertureSize	3
	threshold1	270
	threshold2	400

HoughLines	rho	1
	theta	0.05
	threshold	160
	min theta	0
	max theta	3

HoughCircles	dp	1
	minDist	1
	param1	353
	param2	31
	minRadius	0
	maxRadius	32

Table 2: *Parameters set for road3.jpg image, for all the three algorithms.*

(a) *Original image*

(b) *Edge map after applying Canny to the greyscale image*

(c) *Final result: circle street signs are filled in green, while the area between the intersected lines is filled in magenta*

Figure 3: *Three different steps of the image manipulation: original image in Figure 3a, edge map in Figure 3b and the final result in 3c.*

Canny	apertureSize	3
	threshold1	450
	threshold2	730

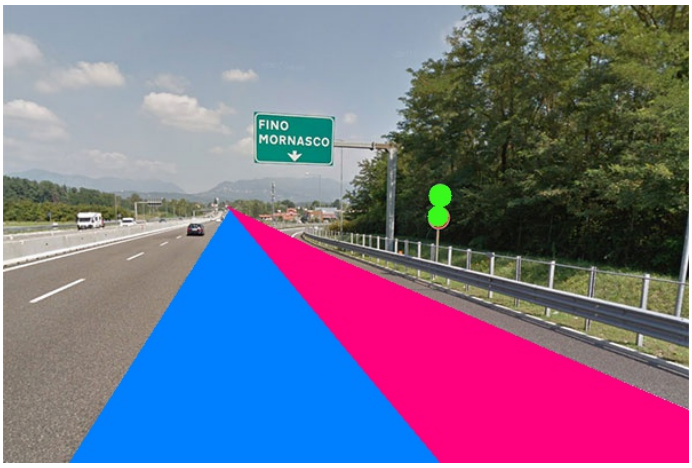
HoughLines	rho	1
	theta	0.05
	threshold	62
	min theta	1
	max theta	3

HoughCircles	dp	1
	minDist	1
	param1	67
	param2	31
	minRadius	0
	maxRadius	14

Table 3: *Parameters set for road4.jpg image, for all the three algorithms.*

(a) *Original image*

(b) *Edge map after applying Canny to the greyscale image*



(c) *Final result: circle street signs are filled in green, while the area between the intersected lines is filled in magenta and blue. Note that in this case we fill differently the two different areas in order to highlight the line in the middle.*

Figure 4: *Three different steps of the image manipulation: original image in Figure 4a, edge map in Figure 4b and the final result in 4c.*