Report Lab expierience 4
Francesco Caldivezzi
ID Number : 2037893

Experience Gained

In this lab I understand how to :

- Use the function **cv::Canny()** that allows me to detect edges in an image
- Use **cv::blur()** to blur an image.
- Use **cv::createTrackbar()** function that allows me to add one trackbar to a windows displayed.
- Use **cv::HoughLines()** function to apply Hough transform to an image to detect lines.
- Use **cv::HoughCircles()** function to apply Hough transform to an image to detect circles.
- Implement my own version of an edge detector, specifically designed for road lines.

Unexpected Issues

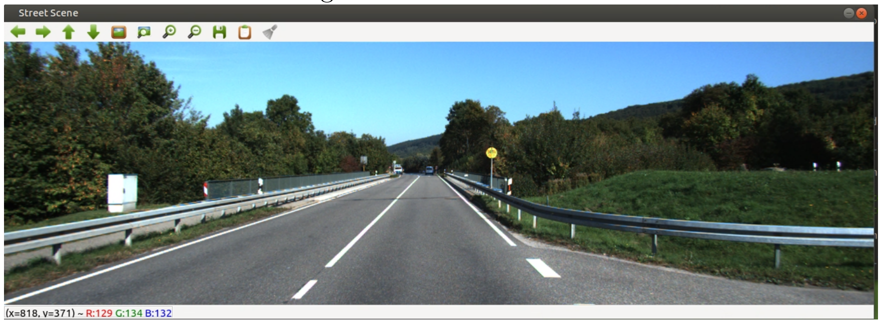
The main difficulties of this Lab were :

- Task1 : the main difficulty here, was to be able to implement the callback function of **cv::createTrackbar()** inside a class. I solved this issue simply by creating a class with static functions one for each callback of track bars that I have added to the window.
- Task2 : the big issue here was in how to implement, actually, a line detector for the lines of the road. I solved this issue by creating a class (RoadLinesDetector) that has a function that apply the following steps :
  - Blurs the given GRAY SCALE image received.
  - Cut off all the pixels below a certain given threshold
  - Compute the Absolute gradient along X axis
  - Cut off all the pixels below a certain given threshold
- Task 3 and Task 4: the main problem in these tasks was to tune the parameters in order to detect the lines in the road with the function **cv::HoughLines()** (Task 3) and to detect the signal in the street with the function **cv::HoughCircles()** (Task 3)
- An additional problem faced with this lab was that, for the first time, I have had an execution error in the virtual machine but not in my local configuration. The problem was that I was passing to the function **cv::createTrackbar()** as third parameter (namely the initial value of the trackbar) simply the value 0. Actually if we observe the signature of the method, the parameter is a pointer to an integer value, therefore we have to declare a variable, assign to it a value and then, passing the reference of it to the function. So, with this change I was able to solve the problem.
  Notice that, a possible explanation why I was able to run my code in my local conf. and instead not in the virtual machine,is that, probably, the compiler of visual studio was implicitly translating such quantities to variables, and so everything was working fine.

Results

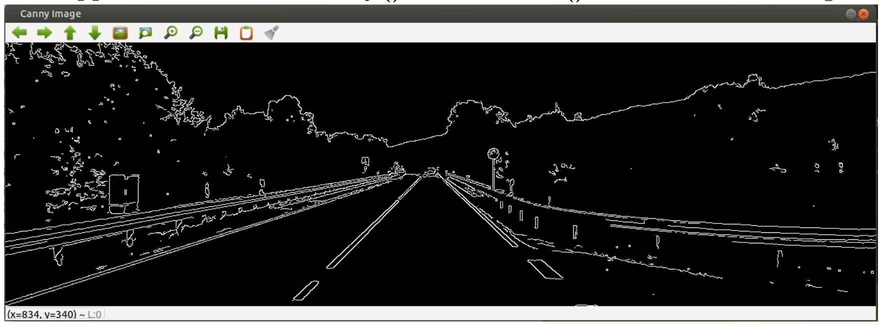
In this section we talk about the experimental results :

- Task1 : The Reference Image:



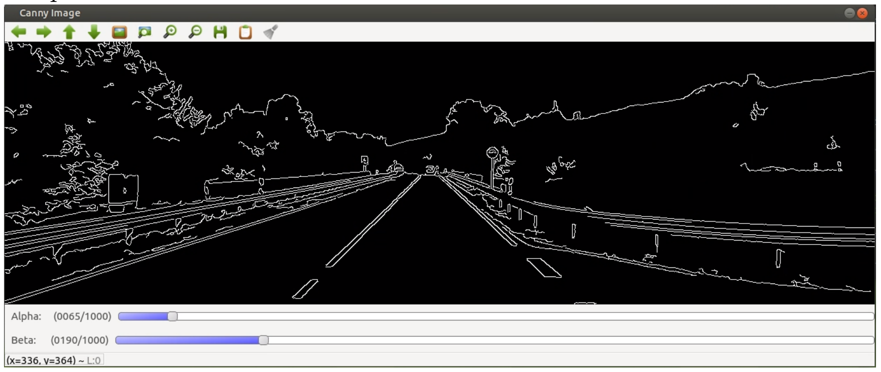
(x=818, y=371) ~ R:129 G:134 B:132

The Application of : **cv::Canny()** and **cv::blur()** to the reference image:



(x=834, y=340) ~ L:0

The Application of : **cv::Canny()** and **cv::blur()** with certain values of the parameters:

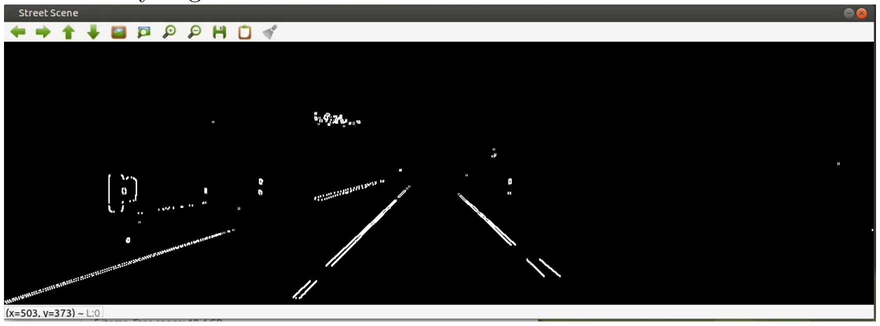


Alpha: (0065/1000)

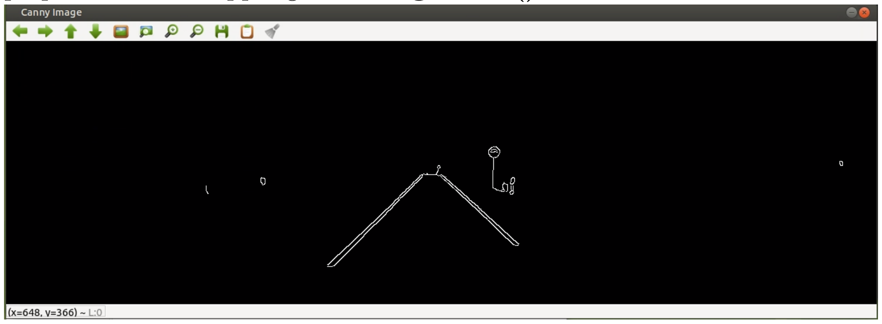
Beta: (0190/1000)

(x=336, y=364) ~ L:0

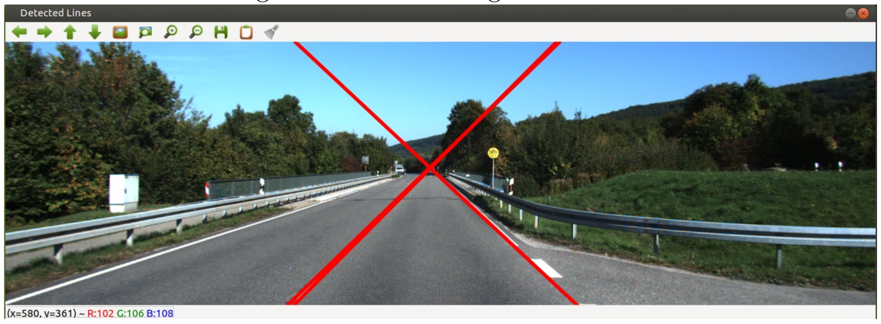
- Task2 :
  Result of my edge detector for the lines of the road:



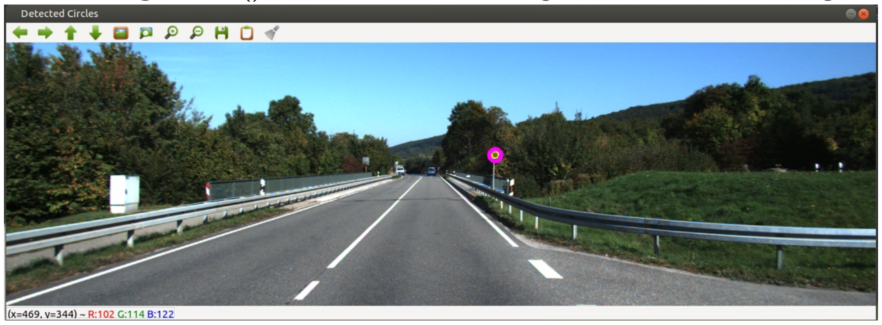
- Task3 :
  Result of the application of **cv::Canny()** and **cv::blur()** as a preprocess before appling **cv::HoughLines()** :



  Application of **cv::HoughLines()** to the previous image, which translates into the original in the following two lines:



- Task4: Application of **cv::HoughLines()** in order to detect the signal in the reference image:


Detected Circles
(x=469, y=344) ~ R:102 G:114 B:122