# Report on Laboratory 4
# Edge and line detection

Computer Vision UniPD

Alessandro Viespoli 2120824

April 2024

# Setup and instructions to run the code

**OS**: Linux (Pop!_OS)
**Enviroment:** CLion
**CMake:** 3.27.8
**OpenCV:** 4.5.4
Each task has its own `main.cpp` and `CMakeList.txt` file, which can be found in a dedicated directory for each task. The image to be load must be in the same folder as the `main.cpp`.
To run the code simply extract the content of the .zip file, enter task's directory and run in the terminal:

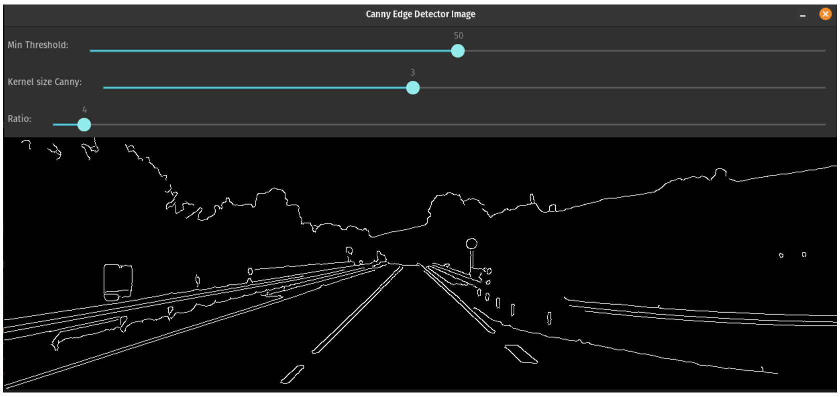
1. cmake .

2. cmake --build .

This will create an executable `task*`; in order to run it pass as `argv[1]` the image file.

# Task 1

For this task I decided to implement 3 trackbars to modify 3 different parameters of the Canny edge detector function: lower threshold, ratio through which multiply the lower threshold in order to obtain the maximum threshold and the kernel size for the Canny edge detector function. A color change and a Gaussian blur are applied beforehand the Canny operation.
All the parameters required both in the main function and the trackbar function are store inside a `struct ParamsCanny`, defined by myself.

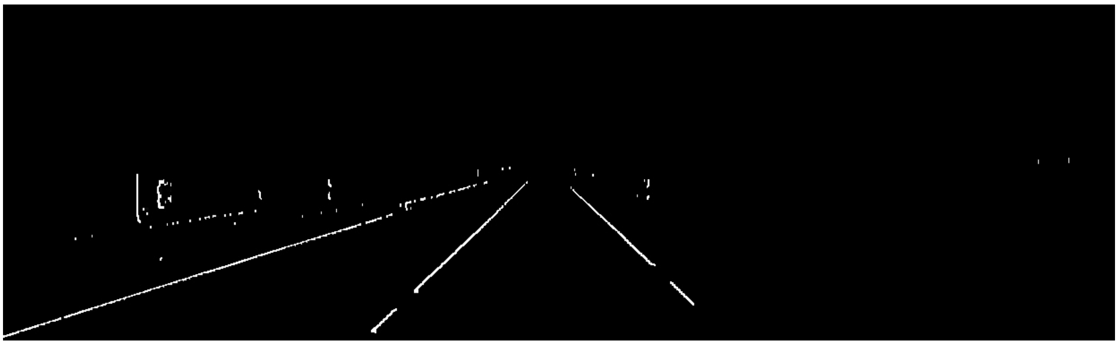


Min Threshold: 50
Kernel size Canny: 3
Ratio: 4

# Task 2

To detect the white markings on the road, I proceeded with the following operations:

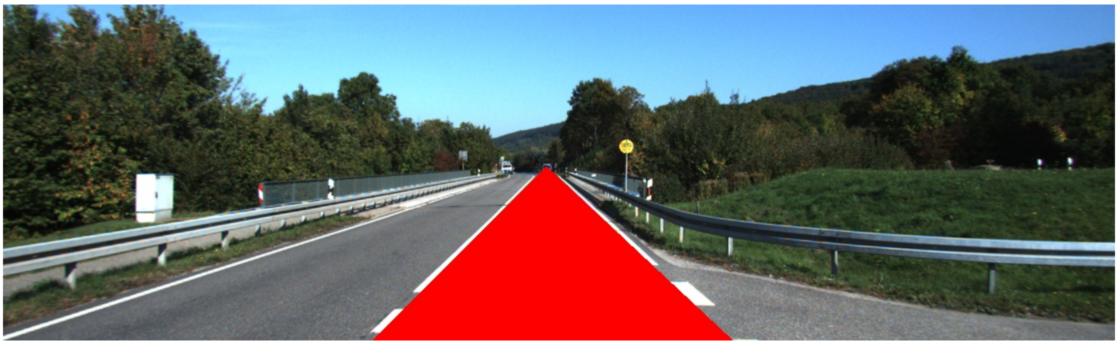
1. Convert the BGR image to grayscale
2. Apply a thresholding by setting to 0 the values below 245
3. Sobel for oriented edge detection (only derivative on the x axis)
4. Closing operation (dilation then erosion).



My flow of thoughts was that since the white marks have a strong light component (when in grayscale), I first isolate them from the rest of the image; then the Sobel operation was used to select only edges with a gradient variation on the x axis. Since the white markings from step 3 were practically a set of disconnected white points, I proceeded with a dilatation followed by erosion (Closing operation). The resulted image correctly shows the white markings, nevertheless there is some noise along the white markings. The method could be improved by appling a kernel for diagonal edge computation on the Sobel function (I couldn't find a way to apply it).

# Task 3

In task 3, I started by first converting the BGR image in gray scale, then a Gaussian blur of size 9x9 had been applied to smooth out the image for a cleaner edge detection. After some parameter tuning (by using the trackbars in task 1), the grayscale image had been passed to the Canny edge detector method. Finally, the `HoughLines` method was exploited in order to select the desired lines and then by extracting the $\rho$ and $\theta$ of the polar coordinates, the color change of the road was just a simply low level manipulation by selecting the area below the two selected lines.



# Task 4

In task 4, I started by first converting the BGR image in gray scale, then a Gaussian blur of size 9x9 was again been applied to smooth out the image. After, the smoothed image had been passed to the `HoughCircles` function and as method parameter I had used the `HOUGH_GRADIENT_ALT`. Since the road sign is very close to a perfect circle, the parameter `parameter2` had been set to 0.9, where 1 is a perfect circle.