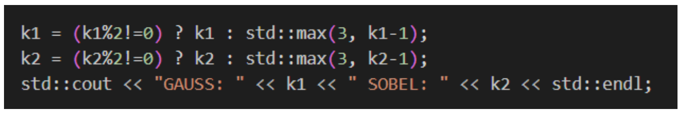
Lab 4 – Edge and line detection

Morselli Alberto (2102413)

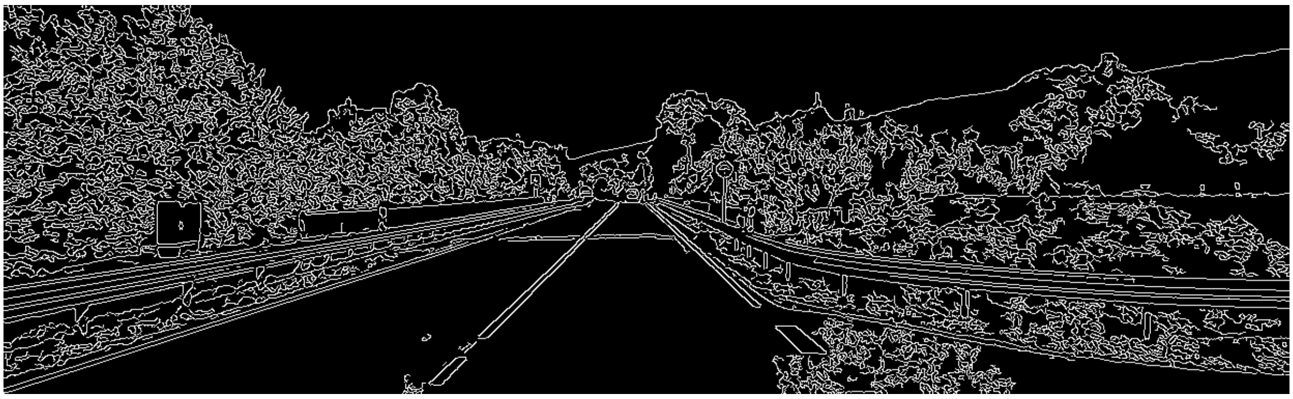
-short report-

≥ ./task1

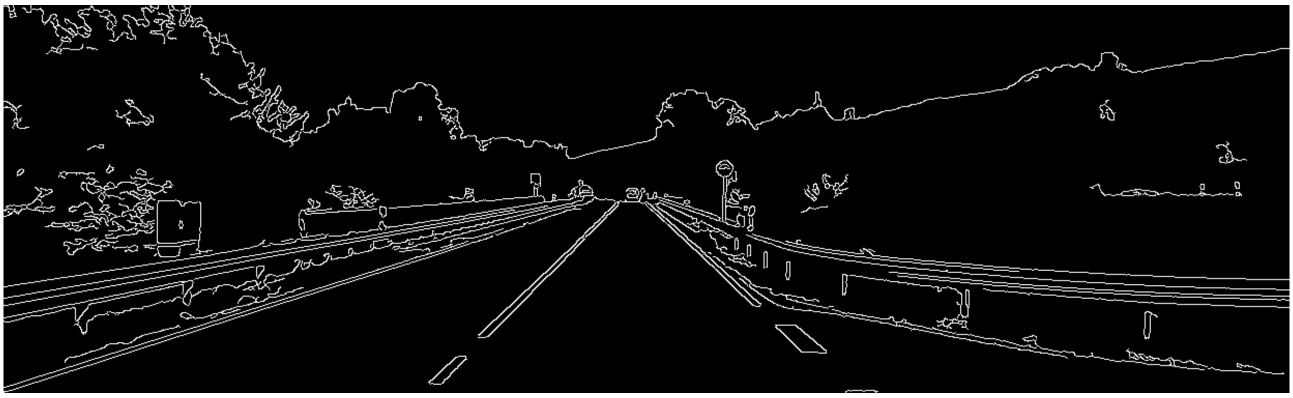
I handled task1 using 4 taskbar: 2 for the low and high tresholds, one for the size of the gaussian kernel filter and one for the size of the sobel filter of canny edge detector. Size of filters required addictional care because of the values that have to be odd. I handled the situation "rounding" the taskbar value to the closest before odd number. Two examples images obtained playing with these 4 values are shown below. With tresholds it's possible to obtain more or less detailed images, while playing with the size of the kernels it' s possible to obtain extravagant patterns.



```
k1 = (k1%2!=0) ? k1 : std::max(3, k1-1);
k2 = (k2%2!=0) ? k2 : std::max(3, k2-1);
std::cout << "GAUSS: " << k1 << " SOBEL: " << k2 << std::endl;
```



more detailed image obtained with tresholds 8-87, gaussian kernel size = 3, sobel kernel size = 3

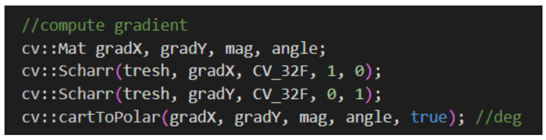


less detailed image obtained with tresholds 46-214, gaussian kernel size = 3, sobel kernel size = 3

≥ ./task2

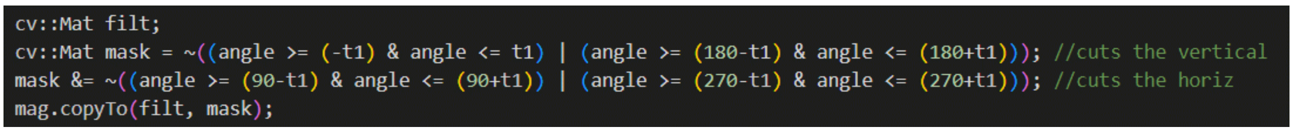
I tried multiple solution for this task, but the "less bad" results were obtained using a 3-stage processing:

1. treshold (with tolerance) on the white color (255,255,255).
2. Derivation of the gradient on the tresholded image by two Scharr filters instead of Sobel ones to trying to achieve better results, but it didn't show noticeable improvements.

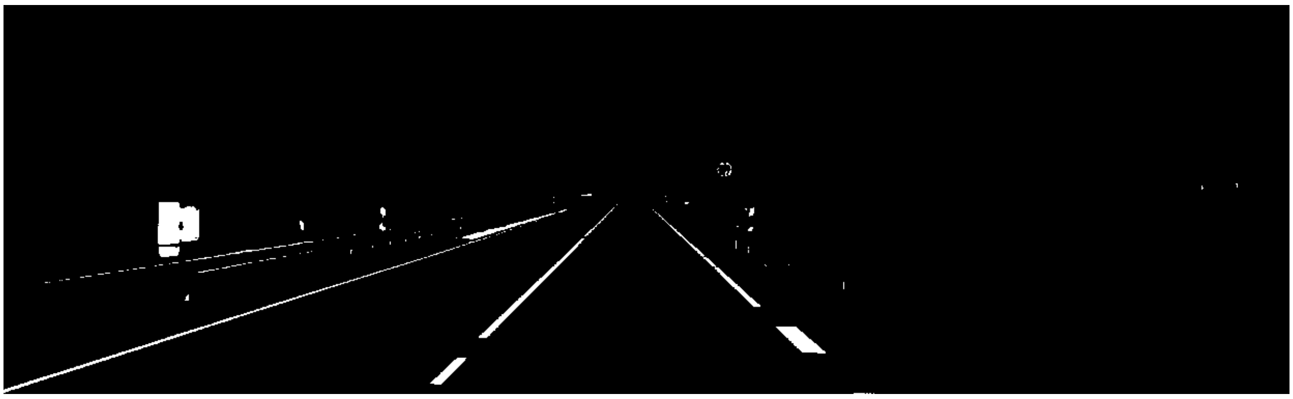


```
//compute gradient
cv::Mat gradX, gradY, mag, angle;
cv::Scharr(tresh, gradX, CV_32F, 1, 0);
cv::Scharr(tresh, gradY, CV_32F, 0, 1);
cv::cartToPolar(gradX, gradY, mag, angle, true); //deg
```

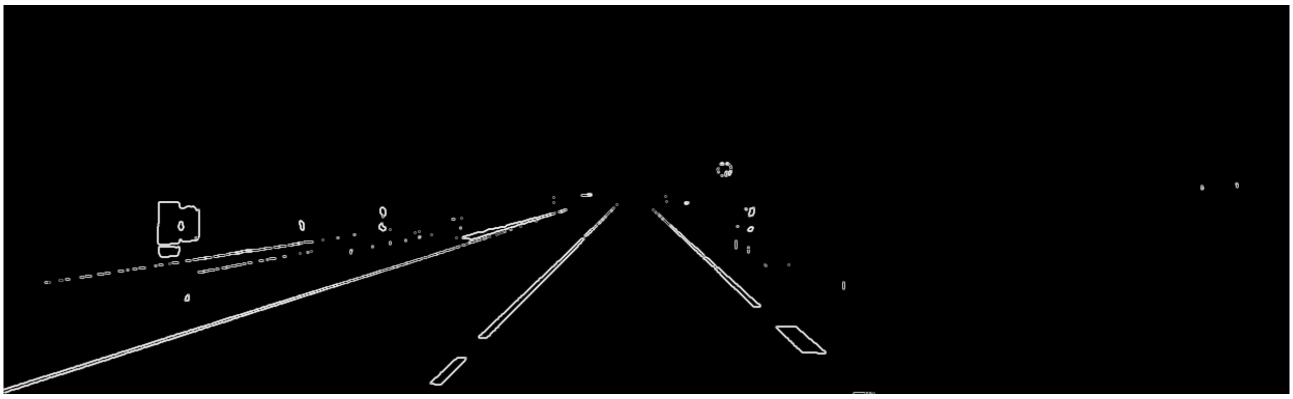
3. Based on the fact that wanted elements (road stripes) are oblique, i then proceed by filtering unwanted elements looking on the pixel angle obtained from gradient and cleaning out vertical and horizontal direction. The three steps are shown on the next page.



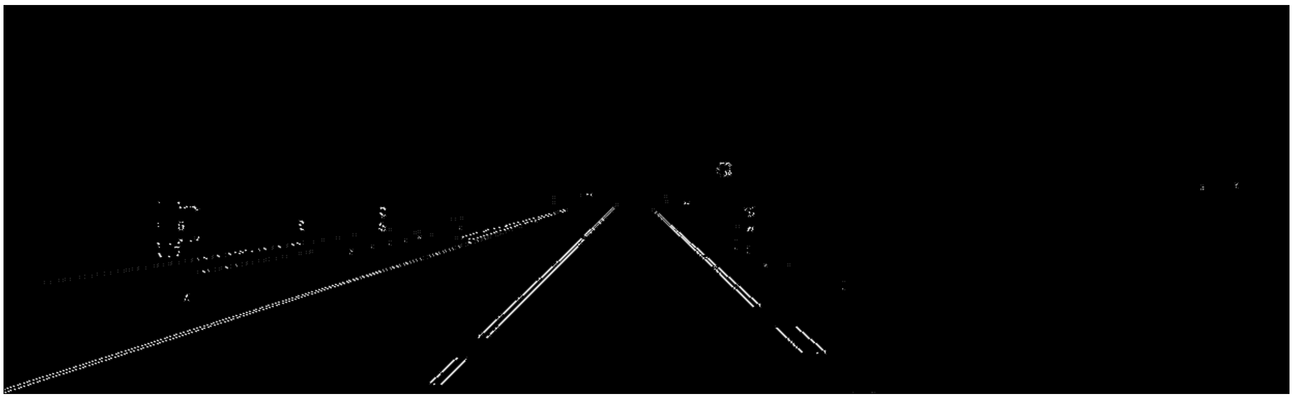
```
cv::Mat filt;
cv::Mat mask = ~((angle >= (-t1) & angle <= t1) | (angle >= (180-t1) & angle <= (180+t1))); //cuts the vertical
mask &= ~((angle >= (90-t1) & angle <= (90+t1)) | (angle >= (270-t1) & angle <= (270+t1))); //cuts the horiz
mag.copyTo(filt, mask);
```



1° step: treshold >= (231,231,231) looking for the white elements



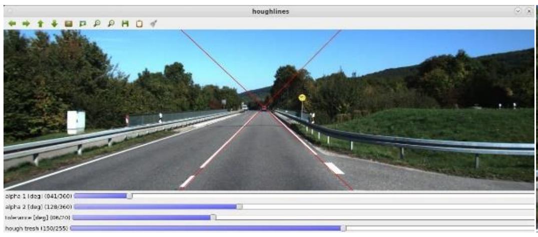
2° step: gradient magnitude and phase computation



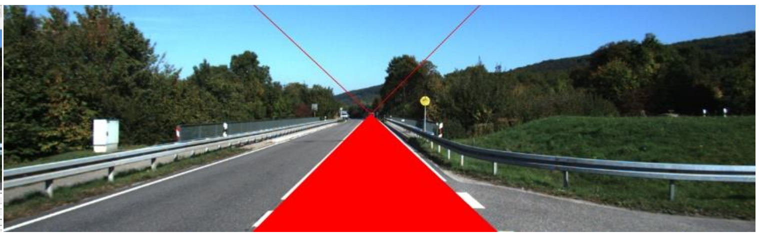
3° step: angle mask filtering out vertical and horizontal element with tol= ±27 [degrees]

≥ ./task3

I handled this task with this theoretical approach: first sorting by lenght [rho] the lines obtained trough hough lines transform. Iterating from the longest then i look for the two desired angles [theta] corresponding to the road stripes. The first one found is surely the wanted one, because we sorted elements by length. The practical approach required some tolerances and some trial-and-error approach looking for the desired angles and the adapt tolerance for getting the require line. Remember that the angle is quantized and sensitive to numerical precisions and 45 degrees could be encoded as 43. I handled the research with some taskbar, once found the "right" parameters of the taskbars, i coloured the triangle obtained on the intersection of the two lines.



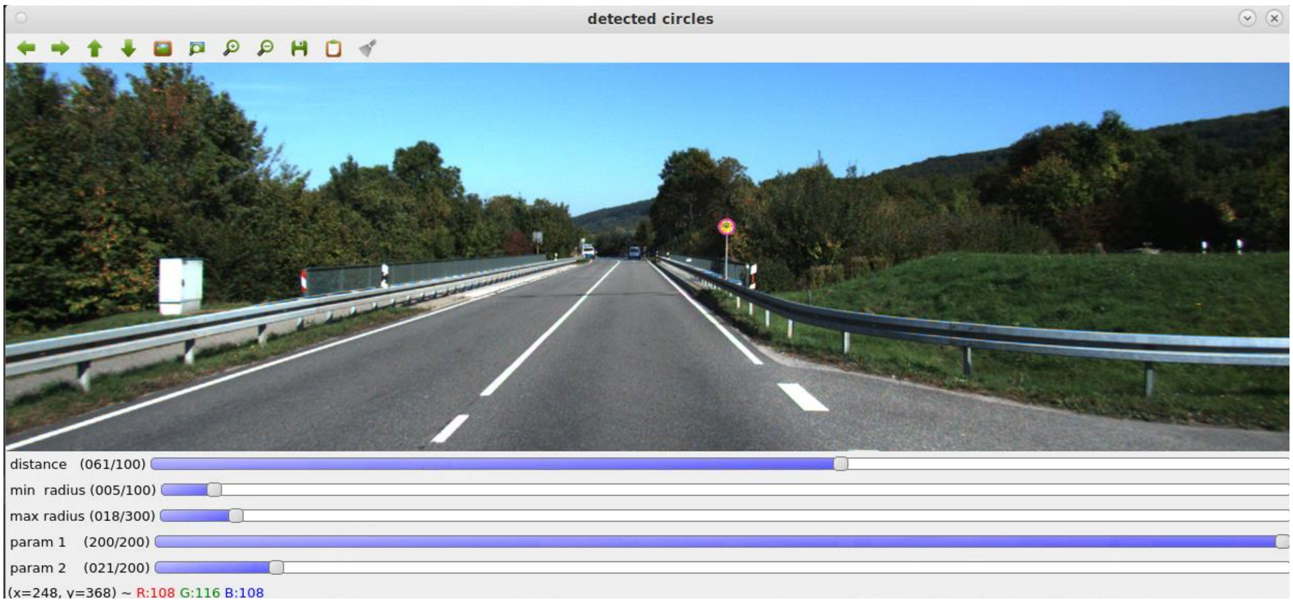
alpha 1 [deg] (041/360)
alpha 2 [deg] (128/360)
tolerance [deg] (06/20)
hough tresh (150/255)



images obtained with alpha1=45 [deg], alpha2=128 [deg], tolerance = ±6 [deg] and hough treshold = 150

≥ ./task4

Results for this task are even more strongly dependent on finding the right way of setting up parameters. The code i designed cares so just to give as more flexibility as possible allowing the use of 5 taskbars to handle the generalized hough transform for circles. The two more important one are min and max radius. I set them measuring the pixel size of the street signal by hand filtering the majority of the other “fake” circles on the image. Also the distance parameters allow to “single out” the circle we desire from the potential other ones. The last two params are also important because allow the detection of the street signal we desire. Being it very small and being the hough transform based on finding the “strongest” ones by treshold parameters, the one we desire is not favoured at all. This require to relax a lot the last two parameters to add it on the list of detected ones, and filtering the unwanted using the first three parameters. The results obtained with the correspondent tresholds are shown below.



```
distance   (061/100)
min  radius (005/100)
max radius (018/300)
param 1    (200/200)
param 2    (021/200)
(x=248, y=368) ~ R:108 G:116 B:108
```