# Edge and line detection

The goal of this lab was to learn about edge and line detection, using standard algorithms like Canny and the Hough transform, but also implementing something new.

Task 1:

- I was able to manage all trackbars with a single callback. This was possible by creating an apposite `struct Params` holding the parameters modifiable by the user, plus the original image. Everything is passed as a pointer, so that each invocation of the callback has access to up-to-date data
- the aperture size parameter was forced into valid ranges, showing an error message to the user, but avoiding any crash
- the two thresholds decide the permissibility of the filter, allowing more or less contours to be defined as lines. Both increasing the aperture size and enabling the L2 gradient, makes the filter less selective

Task 2. Inspired by Canny filter, my filter works as follows:

- smoothen the image (converting to grayscale for convenience)
- apply two diagonal Sobel filters, to detect diagonal edges
- sum the two filters, to consider both directions
- apply a first threshold on pixel intensity, to eliminate the weak edges
- apply a second threshold based on pixel color (similarity to road)

Task 3:

- I applied the `HoughLines` function as suggested in the official documentation (with preprocessing)
- I then selected the appropriate lines by considering two factors: orientation (close to 45 and 135 degrees) and number of votes (intersections in the Hough plane)
- given the two lines, I mathematically computed the three vertices of the triangle

- I then wanted to visualize the Hough plane, so I used the function `ximgproc::FastHoughTransform`

Task 4:

- same as Task 3, but using function `HoughCircles`