I did my task in Python and ran it in Jupyter Notbook. I have also submitted the project on my GitHub account with link of: https://github.com/saman-nia/The-first-assignment-for-IPA

Here I have tried to explain beyond of my codes. Regarding the transformations, histograms is not a rotation-scale invariant descriptor. Typically multi-scale detectors run the train detector at different image scales as OpenCV detector does. However, this will include more complexity and uncertainty in the detector and depending on the objects I was trying to detect it might not be a viable solution.

Regarding the Canny edge detection, I implemented Canny Edge detection algorithm and I think I understood every step of canny edge detection, but when I compared it to results given by OpenCv implementation, I realized that they are different. It seems that I just can't get the 1px wide edges like the algorithm should produce.

When I performed hysteresis thresholding in my implementation, I got very thick edged result. The problem is gradient magnitude values, but I did something to solve it. Actually the gradient magnitude should roughly sketch the edges already. The next steps was refining the edge extraction.

For refining the edge extraction I did interpolation to find the pixels where the ideally of gradient are local maximum. When I applied Sobel operator, I didn't get very thick edges. When I used standard deviation of the Gaussian function to convolve the original image in order to get the gradient, I could finally get clear thin edges.

The smallest distance in image is one pixel and the largest change possible is from white to black, so those would correspond with the largest gradients. But Sobel can work with arbitrary matrices whose min and max values could be well outside of 0 to 1 or 0 to 255.