Computer Vision Assignment Report

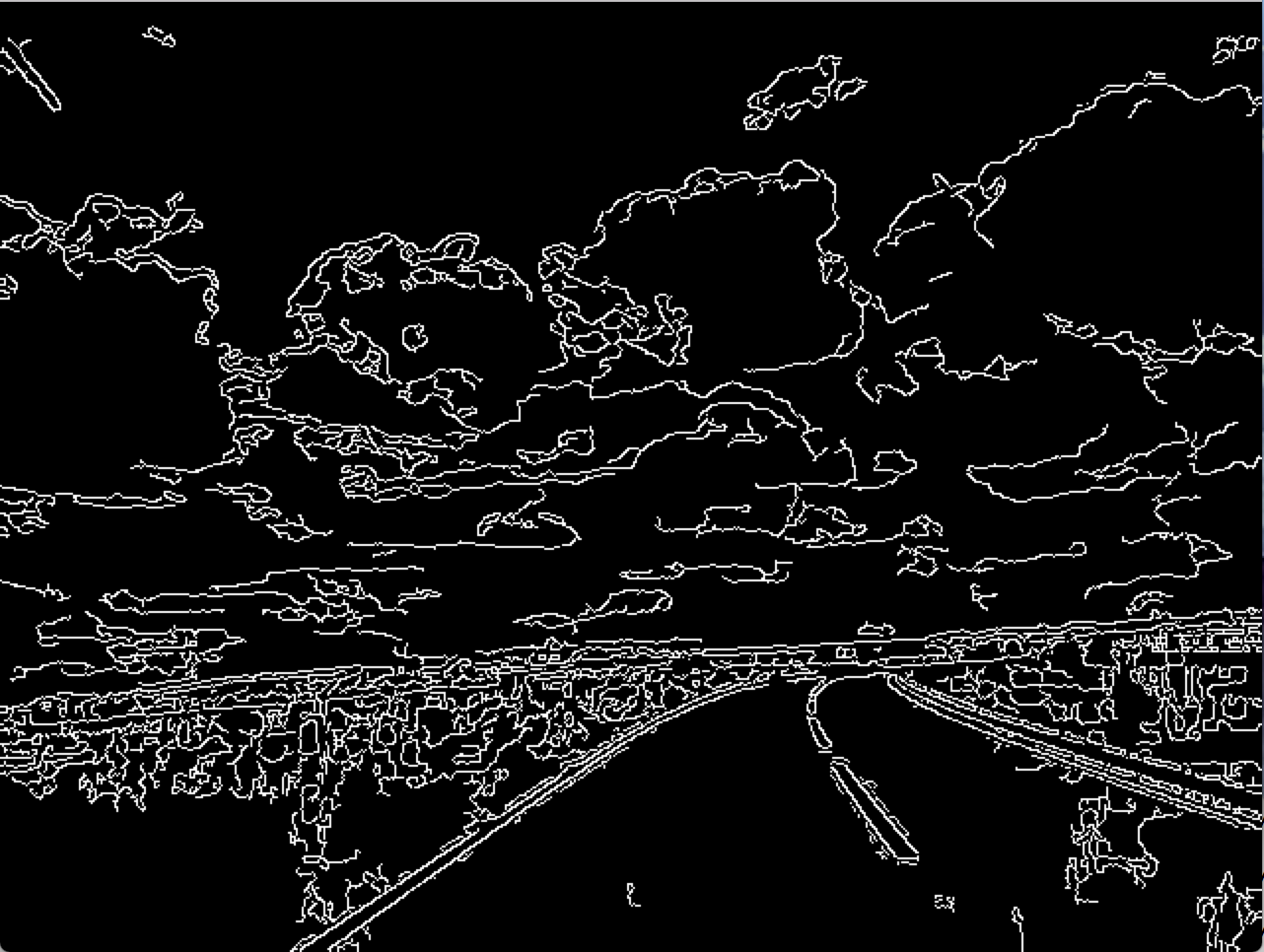
After reading in the image on which analysis is to be performed, the ‘preProcess’ function is called, which performs a variety processing techniques in order to improve the performance of RANSAC as an edge/lines detection method. Firstly, to eliminate variation of edges in the image due to varying illumination the BGR image is transformed into its HSV equivalent, then the three channels are isolated and we extract the saturation component for use in the remainder of the algorithm. This ensures that we still have the characteristics of the image but they are not influenced by illumination.



Illumination invariant transform

Original Image (vlcsnap-00400.png)

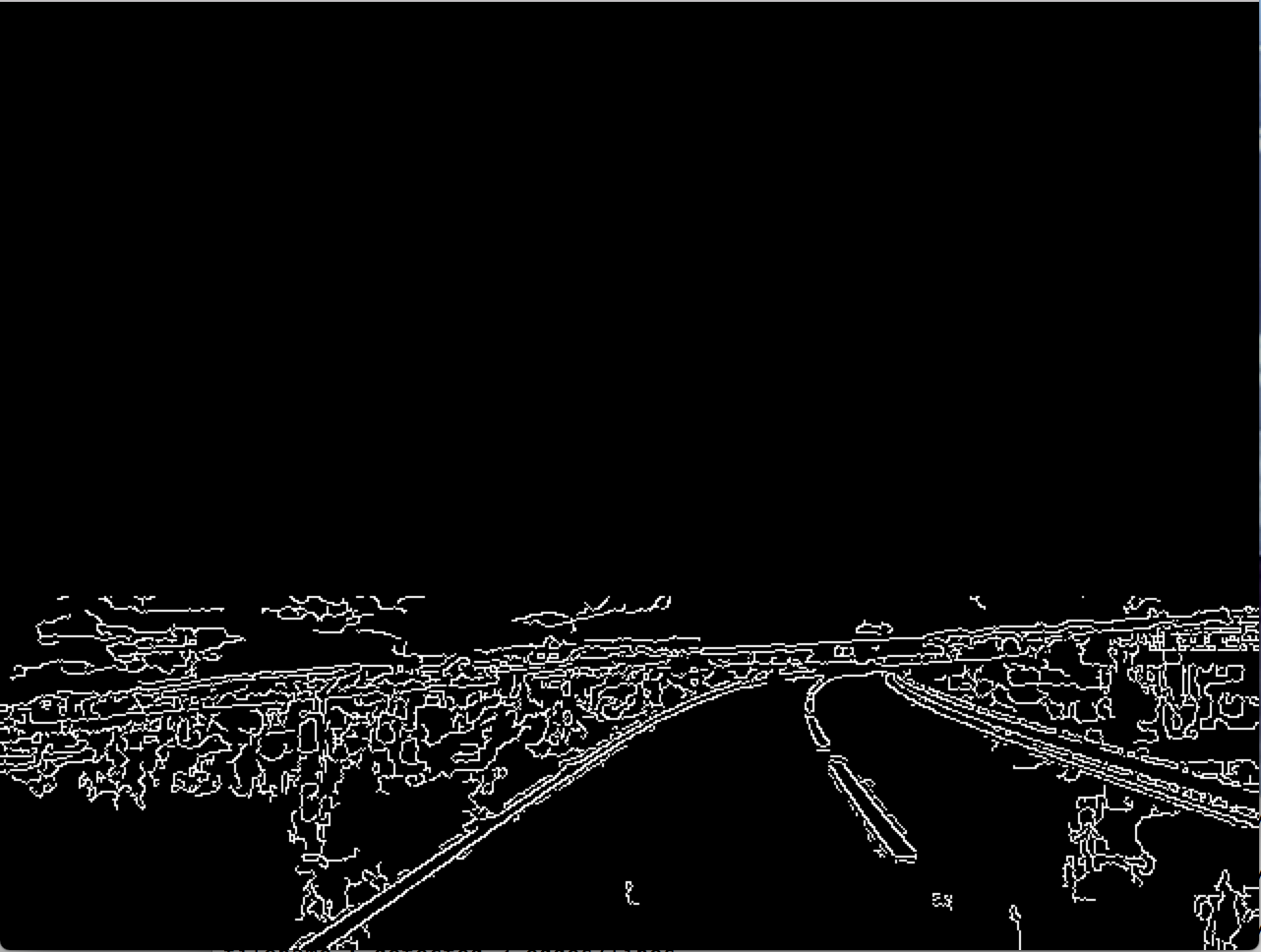
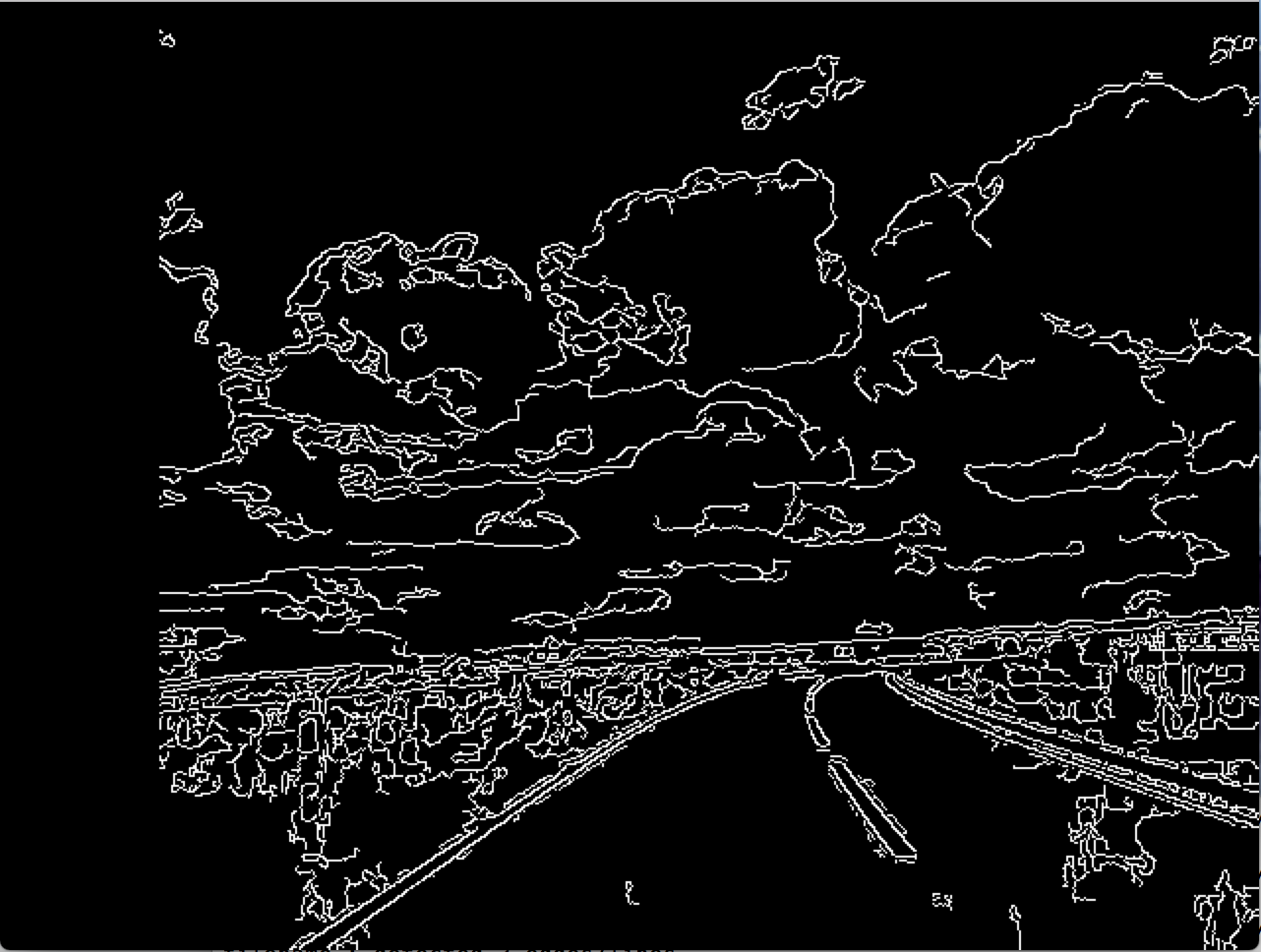
Then, Gaussian blur is applied to the S component of the image, which helps us to blur insignificant features and aids in the extraction of edges. A canny edge detector is then applied to this image, which performs the sophisticated canny algorithm in order to optimally detect edges within the image.



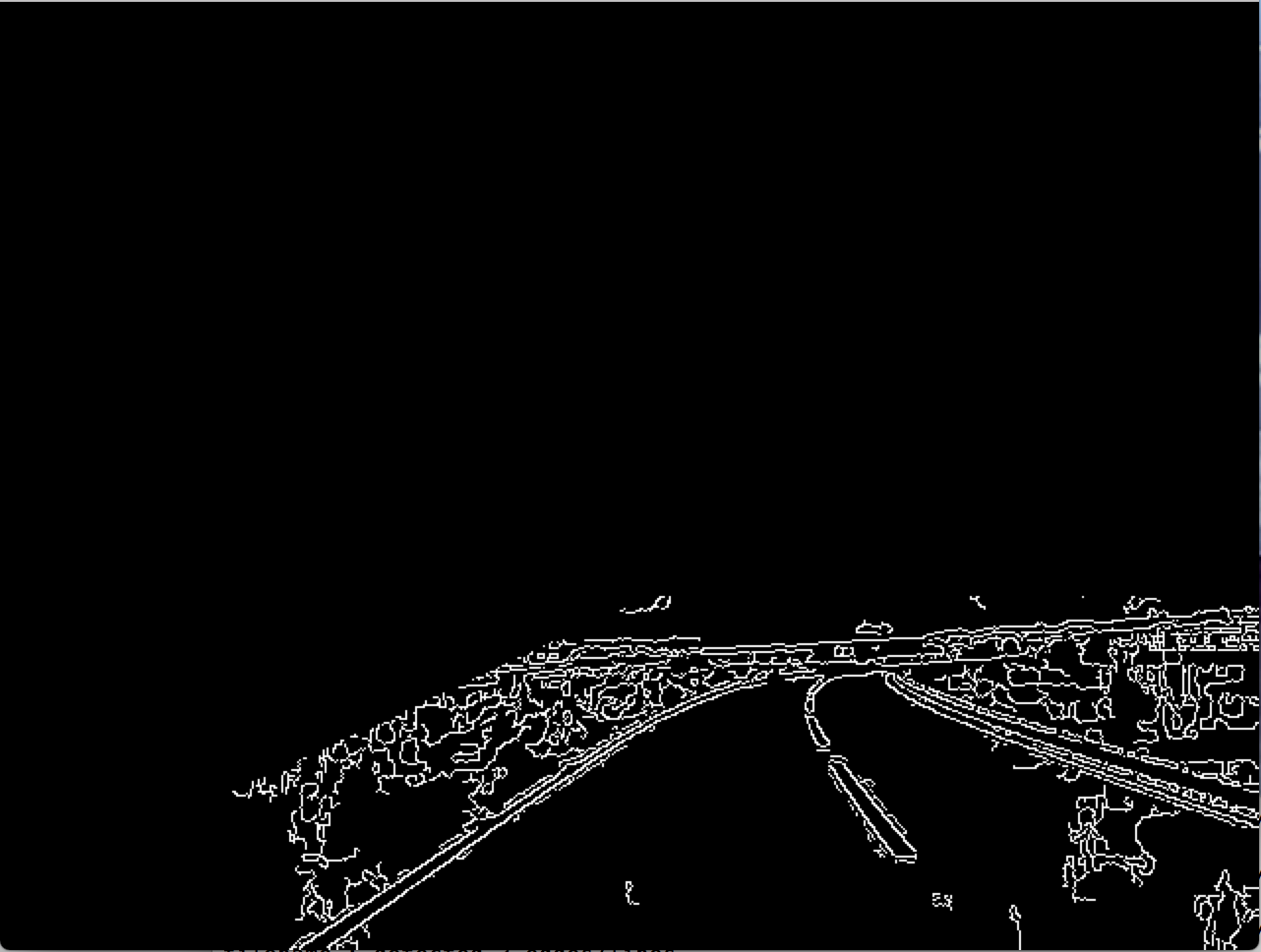
Canny edge detection

Gaussian blurred image

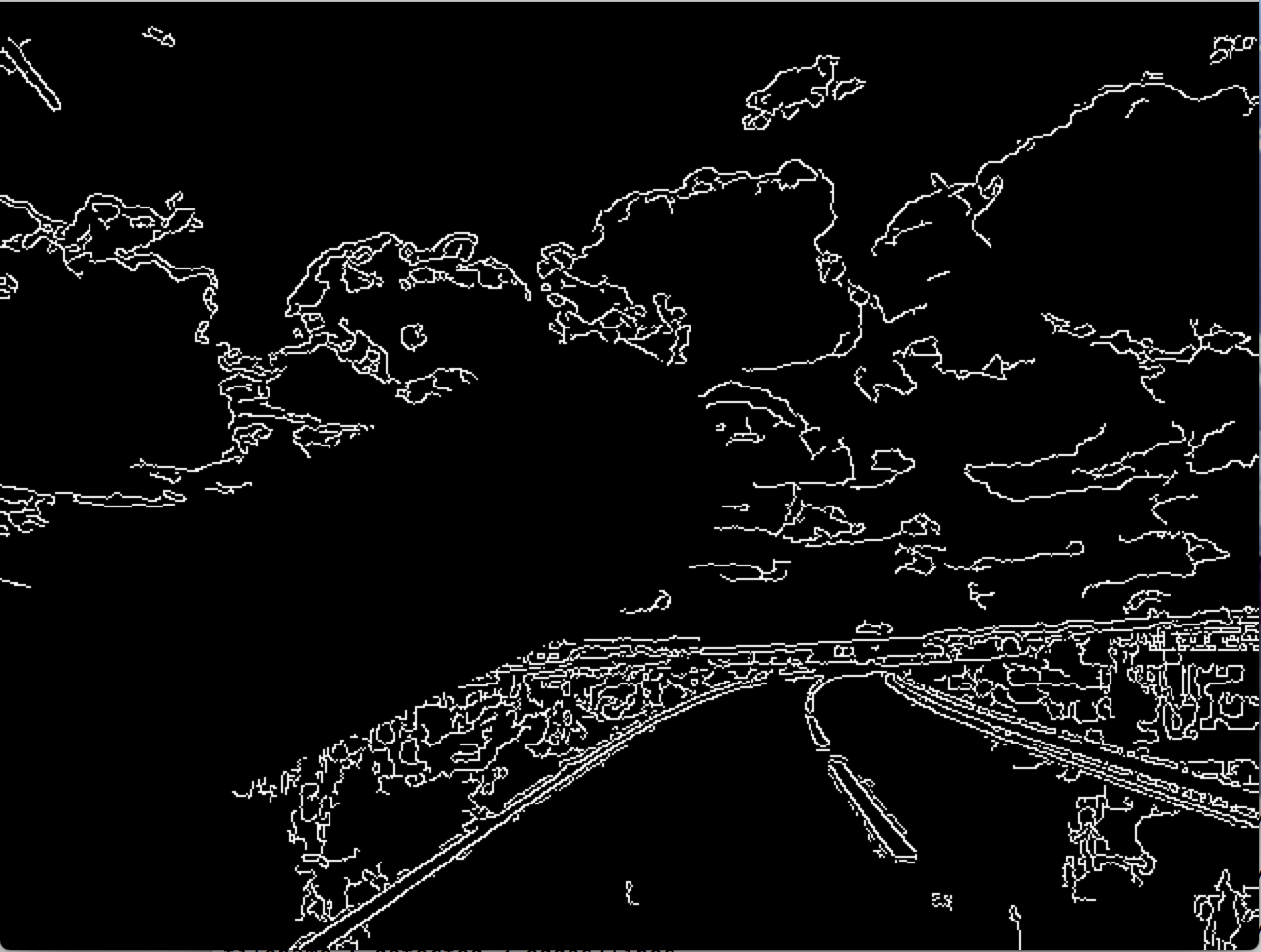
Finally, some heuristics are applied which aid in the detection of road edges/markings: A black rectangle is drawn upon the image of edges using the OpenCV rectangle function. This rectangle is drawn using co-ordinates (0,0) and (cols,int(5\*rows/8)) (where rows and columns are the dimensions of the image i.e. rows,cols = image.shape) in order to eliminate any edges in the top of the image (above the road) from consideration by our RANSAC algorithm. This reduces false detection and computation time. Similarly, a black rectangle is drawn using co-ordinates (0,0) and (int(cols/8),rows), this draws a rectangle which obscures the edges in the far left section of the image. This also aids in a reduction of false detection and computation time because we know that no parts of the road will feature in this section as the car drives on the left-hand side of the road (this could be changed if the algorithm was to be used internationally). To further aid in this matter, a thick line is drawn on the left side of the image, roughly parallel to where any road edges would be, in order to block any more road-side unnecessary edges.



Heuristic Two



Heuristic One



Final pre-processed image

Heuristic Three

Then, we perform the RANSAC algorithm. The algorithm uses the preprocessed image obtained from the function described above. Firstly, the white (edge) pixels are extracted and put into a separate array, so we can reference them. We now perform RANSAC to find road edges/markings until the number of inliers for the respective line found is less than half of the number of inliers of the first line found. Two of these white pixels are randomly picked to be candidates for a line in each iteration of the algorithm. However, a heuristic has been added here to ensure that we only obtain reasonable candidate lines. To eliminate vertical and horizontal lines being chosen as candidates we check that the difference between the x co-ordinates of the two points is at least 50, and the difference between the y co-ordinates is at least 25. This reduces false detection rate due to things like horizon lines and lampposts / vertical lines due to glare. To check the strength of the candidate line, we create a blank image of the same size as the input image and draw a white line between the two co-ordinates. We then logically AND this line image with the processed image and obtain the inliers to the line. The inliers are counted and if there are more of them than the best line so far, we update the best and remember this line. This process repeats until we have performed 10,000 line calculations. Each time we find a best line, we call a function ‘extendLine’ which checks whether we can extend the line in either direction to better model the entire length of the line. This is done by extending the line in small increments and checking if the number of inliers to the line increases significantly with the increase in length. The resulting line is returned back to the RANSAC function and the red line is drawn on the output image.