Jrnf24

Computer Vision Assignment Report

Algorithm Outline

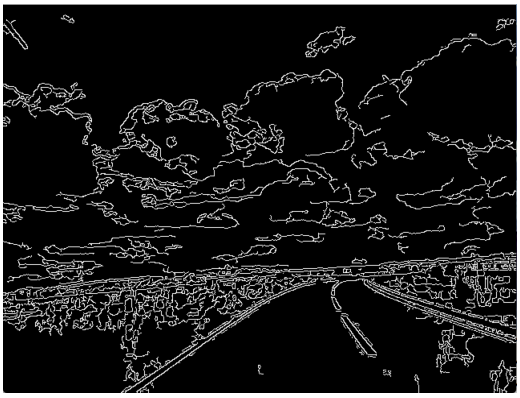
After reading in the image on which analysis is to be performed, the 'preProcess' function is called, which performs a variety processing techniques in order to improve the performance of the RANSAC algorithm. Firstly, to eliminate variation of edges in the image due to varying illumination the BGR image is transformed into its HSV equivalent, then the three channels are isolated and if the average of the V component is less than 125 we extract the saturation component for use in the remainder of the algorithm, otherwise we just transform the BGR image into grayscale. This ensures that we have the characteristics of the image but they are not influenced by illumination.

Original Image (vlcsnap-00400.png)

Illumination invariant transform

Then, Gaussian blur is applied to this image, which helps us to blur insignificant features and aids in the extraction of edges. A canny edge detector is then applied, which performs the sophisticated canny algorithm in order to optimally detect edges within the image. We then count the number of edge pixels, if this is greater than 30000 we change the parameters of the canny filter to be more strict in order to remove more unnecessary edges from the image.

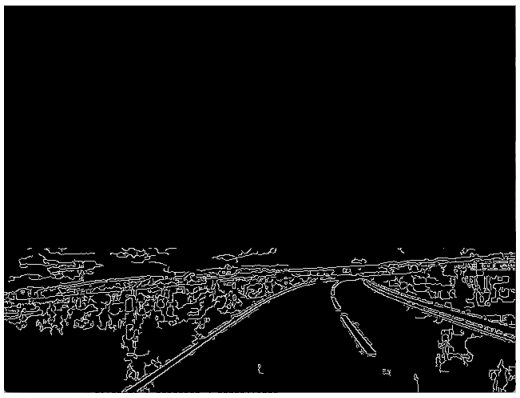
Gaussian blurred image



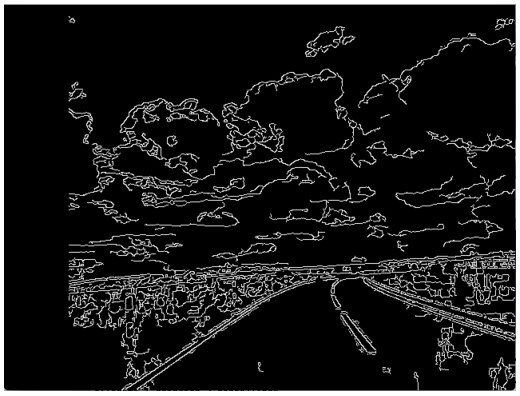
Canny edge detection

Finally, some heuristics are applied which aid in the detection of road edges/markings: A black rectangle is drawn upon the top image of edges using the OpenCV rectangle function 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 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 unnecessary road-side edges.

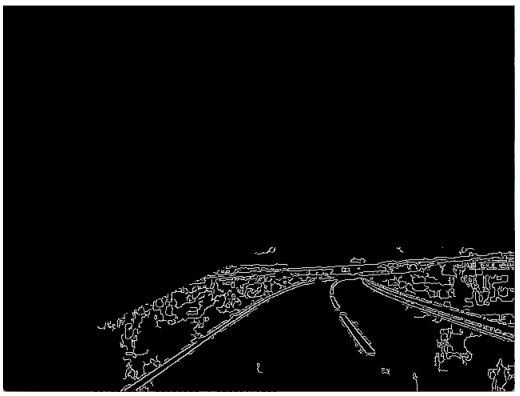


Heuristic One



Heuristic Two

Heuristic Three



Final pre-processed image

Then, we perform the RANSAC algorithm upon the processed image. 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 until the number of inliers for the respective line found is less than half of the number of inliers of the first line found, or we have 4 lines, or there are less than 1000 edge pixels remaining. Two of these white pixels are randomly picked to be candidates for a line in each iteration of the algorithm. A heuristic has been added here to eliminate vertical and horizontal lines being chosen as candidates by checking that the difference between the x co-ordinates and between the y co-ordinates satisfies certain conditions. This reduces false detection rate due to things like horizon lines and lampposts / vertical lines due to glare.

With point selecting heuristics (vlcsnap-00451.png)

Without point selecting heuristics

With point selecting heuristics (vlcsnap-00328.png)

Without point selecting heuristics

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, then logically AND this image with the processed image to 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.

Final RANSAC output (vlcsnap-00500.png)

We can see that in the image above the algorithm outputs a perfect result.

Jrnf24

Statistical Results

Only a subset of the total data set was used because most of the images are members of a sequence of images taken in quick succession.

The Hough transform was performed without any of the above heuristics to ensure results were produced, lines had to manually checked.

Image	Ground Truth	RANSAC	Hough
vlcsnap-00003.png	3	2	2
vlcsnap-00005.png	3	2	0
vlcsnap-00015.png	3	0	3
vlcsnap-00023.png	3	3	1
vlcsnap-00026.png	2	2	0
vlcsnap-00034.png	3	0 (2 false)	0
vlcsnap-00045.png	3	2	1
vlcsnap-00054.png	4	2	2
vlcsnap-00087.png	3	2	1
vlcsnap-00093.png	4	2	1
vlcsnap-00108.png	3	1	0
vlcsnap-00121.png	4	2	3
vlcsnap-00135.png	3	1	0
vlcsnap-00148.png	3	3	0
vlcsnap-00150.png	3	1 (2 false)	0
vlcsnap-00158.png	3	1	0
vlcsnap-00171.png	3	0	2
vlcsnap-00175.png	3	0 (3 false)	0
vlcsnap-00187.png	3	0 (3 false)	2
vlcsnap-00213.png	2	2 (2 false)	1
vlcsnap-00220.png	2	1	0
vlcsnap-00233.png	3	3 (1 false)	0
vlcsnap-00260.png	4	2	2
vlcsnap-00292.png	3	1	1
vlcsnap-00322.png	3	2	2
vlcsnap-00328.png	4	2	1
vlcsnap-00385.png	3	2	2
vlcsnap-00393.png	2	2	2
vlcsnap-00418.png	2	2 (1 false)	0
vlcsnap-00433.png	2	2	1
vlcsnap-00446.png	2	0 (2 false)	2
vlcsnap-00451.png	3	3	1
vlcsnap-00461.png	2	2	0
vlcsnap-00470.png	2	1	1
vlcsnap-00480.png	3	1	1
vlcsnap-00488.png	4	1	1
vlcsnap-00499.png	3	2	2

vlcsnap-00509.png	4	3 (1 false)	2
vlcsnap-00524.png	3	0 (3 false)	2

	RANSAC	**HOUGH**
100% detection	25.64%	7.69%
>= 50% detection	64.1%	43.59%
False positive rate	25%	N/A

From the table above we can see that this RANSAC algorithm isn't a good candidate for performing road edge detection, the detection rate is simply too low. The algorithm struggles greatly when there aren't clear boundaries to the road (vlcsnap-00187.png), when there is poor illumination (vlcsnap-00446.png) or when the road edges are horizontal (vlcsnap00175.png). However, we see that in this case it outperforms the built-in Hough transform. This prototype does show some promise for success in further development, but little.