ENPM 673

PROJECT 2

March 12, 2020

Markose Jacob 117000269
Karan Sutradha 117037272
Trevian Jenkins 116781381
University of Maryland

Problem 1

The aim was to improve the quality of the video sequence. The first step was to convert the video to frames and then apply a filter to reduce the noise in the video. After application of a Gaussian filter, we knew that the video was dark, and we must improve the brightness and contrast in order to have a higher-quality video. So, the next step was to perform a histogram equalization.

Histogram Equalization

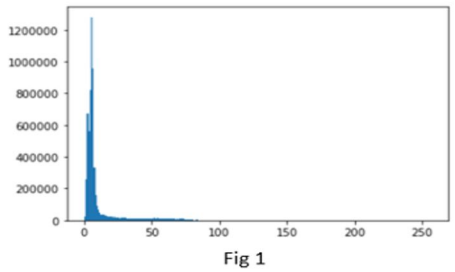

Figure 1: Original video histogram

Figure 2: Original video screen shot

From the plot in Fig 1, we see that the histogram lies primarily in a darker region. We need a uniform distribution of both the dark and bright regions. We must transform the histogram of the image so that it achieves this even distribution. Histogram equalization improves the contrast and brightness of the image.

Grayscale Image - Histogram Equalization

First the frames were read and converted into grayscale. We normalized the video frame using histogram equalization. The idea was to spread out the histogram so that it makes full use of the dynamic range of the image. We got a normalized graph, but the video was rather noisy.

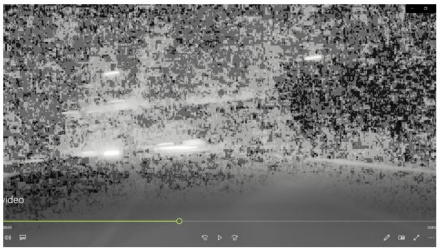

Figure 3: Gray scale Histogram Equalization video output screenshot

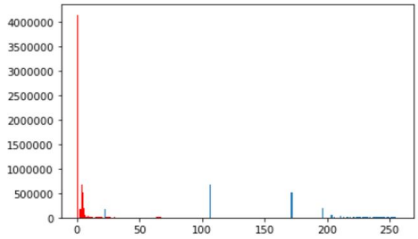

Figure 4: Gray scale video histogram equalization

Canny Edge Detection – Histogram Equalization

In both the above cases the video output was noisy, so we tried application of canny edge detection and then applied histogram. We still got an output with noise.

Color Image - Histogram Equalization

Since the video was dark with color pixels (RGB) present, we separated the images into green, blue and red channels, applied a histogram equalization individually on these channels, and merged the image back. We again got a normalized graph, but the video was with full of noise.

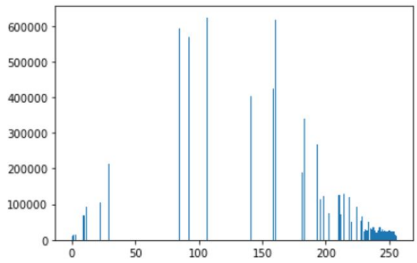

Figure 5: Colour video Histogram Equalization

Figure 6: Colour Video Screenshot

Gamma-Correction

To apply gamma correction, the image pixel intensity is to be scaled from the limit of [0, 255] to [0, 1]. By applying the equation $O = I^{(1/G)}$ we get the gamma-corrected image.
'I' is the input image and 'G' is the gamma value. The output image 'O' is then scaled back to the range [0, 255].
Gamma values less than 1.0 will shift the image towards the darker region and gamma value greater than than 1.0 will make the image brighter. A gamma value of 1.0 will have no effect on the input image. We had the best output after applying gamma correction. In our program we tried using different gamma values and found that the value of 3.0 gives the best result.
We concluded that gamma correction gives the best output in order to make the image brighter and hence discarded our previous attempts to in making the video brighter.

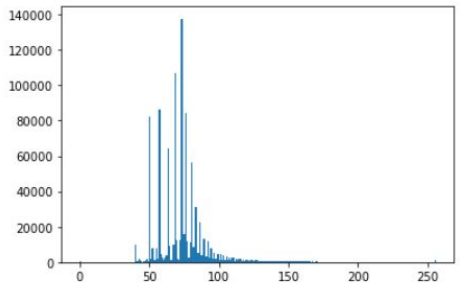

Figure 7: Gamma Correction Histogram

Figure 8: Gamma Correction Video Screenshot

Problem 2

The pipeline we followed to solve this problem is described in this section.

Undistort the image

By using the camera matrix and distortion coefficients we first undistorted the image.

Figure 9: Undistored image

Finding yellow regions which correspond to lanes

We first isolated the yellow regions in the image based on the hue and value, using erosion and dilation to smooth the regions for edge and line detection.

Figure 10: Emphasized Yellow Region

Finding white regions which correspond to lanes

To find the white regions from the masked image we first convert the image to gray scale and use Gaussian and bilateral filters to remove noise, followed be a threshold application. Erosion and dilation are performed again to encompass the lines of interest, and the resulting mask is applied to the original image. This step would find white lines on both sides of the road in case there were no yellow lines.

Figure 11: white regions

Finding gray regions which correspond to surface of the road

The gray region of the image was found similarly to the yellow region. We looked for areas with moderate intensity and low saturation, corresponding to the road. We dilated this area to encompass the region of interest so that background elements (sky, trees, cars, etc.) would be excluded while our lane lines would be preserved. By dilating and inverting this region and using a bitwise-AND operation, we excluded the middle of the road from land detection, overcoming the challenge of distinguishing cracks in the road from lanes, as seen in the challenge video.

Figure 12: gray regions

Edge detection

We then used canny edge detection to find the edges. After edge detection we cropped the image to get rid of the top half where there are no lanes present, and we applied the mask described in the previous section.

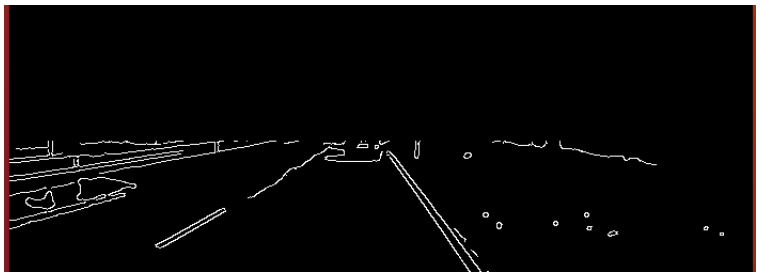

Figure 13: Edge detection

Hough lines

We then used the Hough transform to identify lines within the image. The resulting lines in the Hough transform were separated into two groups based off the slope of each line. Lines with a low absolute slope were excluded, as they more likely arose from background elements. After averaging the slope and y-intercept of each group, we were able to distinguish two major lines corresponding to our lane edges.

Overlay

We found the horizon based off the proportion width of the gray region of the image to place the cutoff for drawing the lane. The lane lines determined from the Hough transform were used to project the lanes onto the original image.

Turn Prediction

Using the four vertices of the Hough lines, we calculated the homography and unwarped the image. Using the yellow and white masks generated previously, we fit a 2nd-order polynomial equation to each line. The sign of the highest-order coefficient determined the 2nd derivative of the line, which in turn determined if the lane was turning, and which direction. If the sign of this coefficient was the same for both sides, then the lane was indeed turning. A positive 2nd derivative corresponded to a right turn, and a negative sign corresponded with the left.

Figure 14: Turn prediction demonstration

Video Output

The video output for problem 1 can be found in the link below :
https://drive.google.com/file/d/1xVnW3HPCQgHPMpyd9vPtDBQxUYa2TQEI/view?usp=sharing

The video output for problem 2 can be found in the link below :

Day Drive: https://drive.google.com/open?id=1rCX3lqPK5dk8fWKKc79EiC4bV_fdv5ed

Challenge: https://drive.google.com/open?id=10ieWuzF8WZSwEkLLht0K0qhOLmLJqB8c
Extra example: https://drive.google.com/open?id=1CQWk95V5ops7hOQCqZ205Dq00V4p3SmG
We recommend to use VLC for playing the videos.

NOTE:
To run the code you need to follow the README file.

Additional

Additionally we also executed our code on different videos and found the results to be satisfactory.

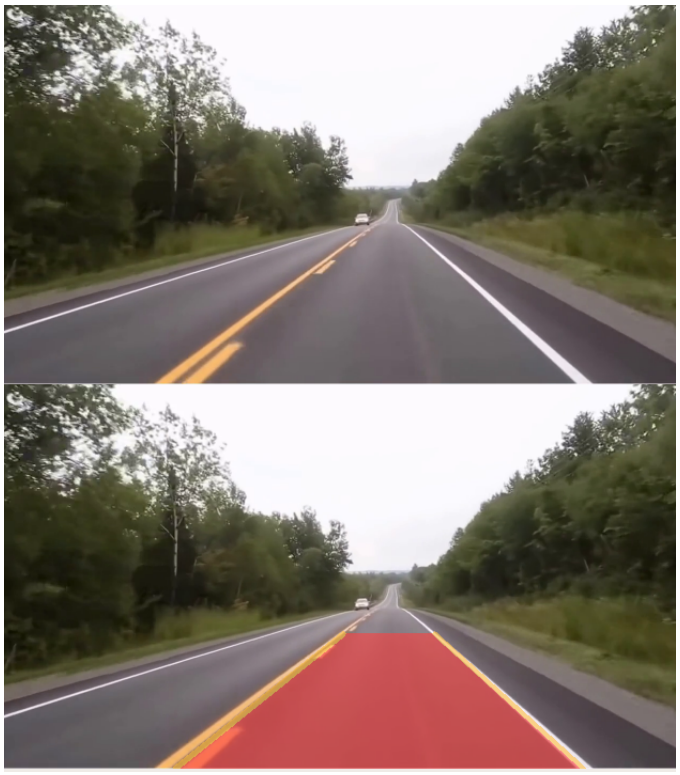

Figure 15: Video 1

Figure 16: Video 2

Figure 17: Video 3