1 Problem1

In the first problem, we are supposed to enhance the quality of the given input video.

1.1 Method Employed

1. First we converted the image into gray and analyzed the pixel distribution using histogram and got the following output.

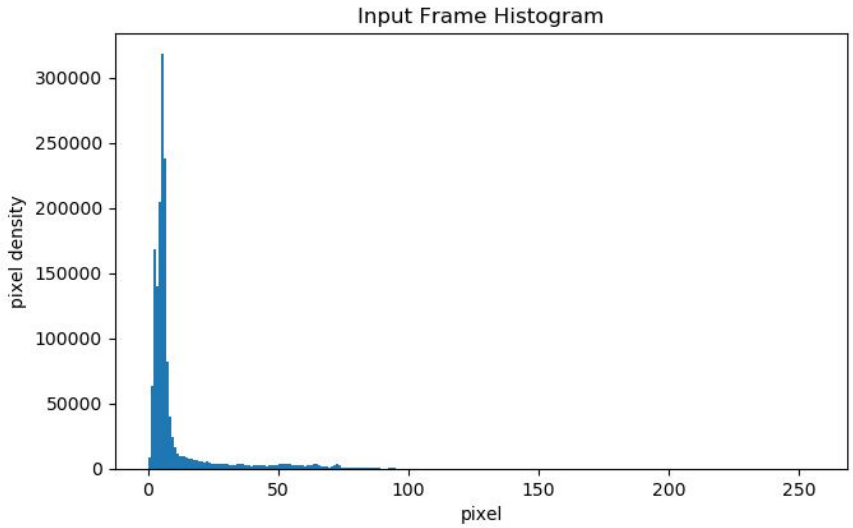

Figure 1: Input frame Histogram output

We saw high density of pixels within the range 0-50. The goal was to increase the contrast by distributing the pixel density evenly between 0-255.

2. Initially we tried to improve the quality of the video using histogram equalization. The output certainly increased the brightness of the video but at the same time amplified also noise.

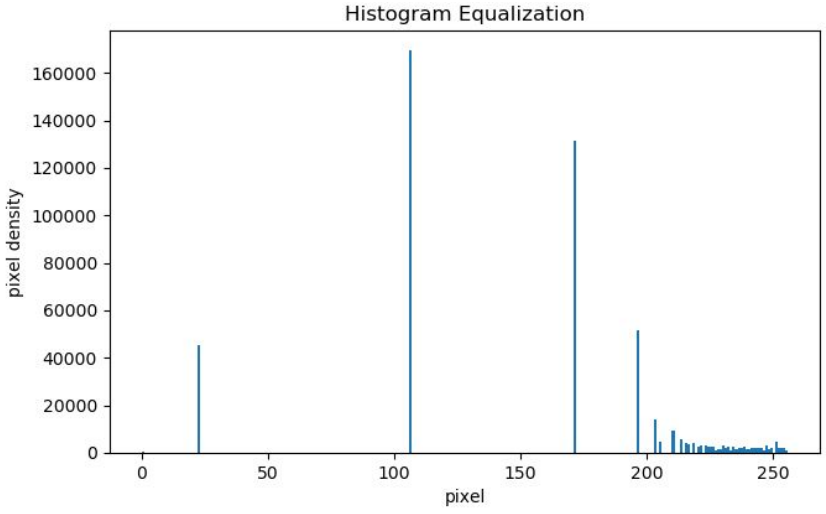

Figure 2: Histogram Equalization output

After performing histogram equalization, the pixel density was distributed but resulted in noise amplification clearly seen between pixel range 100-150. We tried to decrease the noise by using filters but could not decrease the noise.

3. We employed gamma correction. This method could increase the quality by could not increase the contrast.

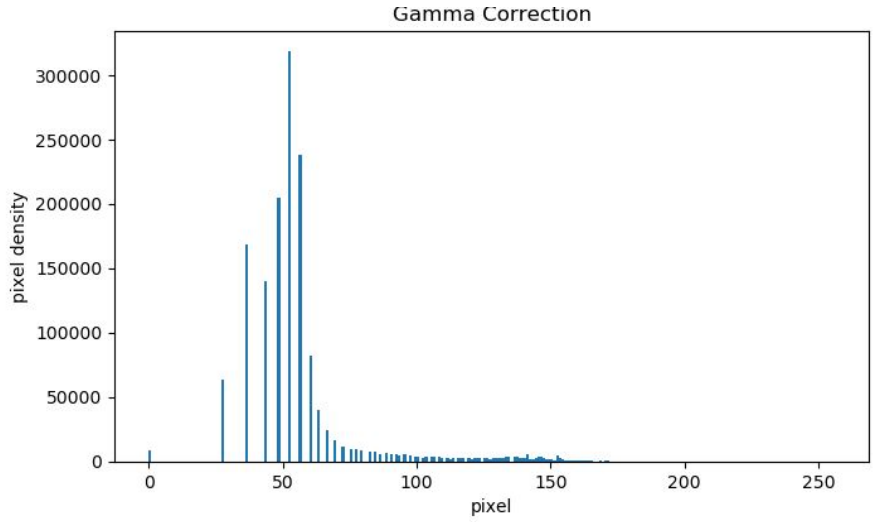

Figure 3: Gamma Correction output

4. Then we multiplied each pixel with α and added by β. We found this method increased the contrast while decreasing the noise compared to histogram equalization. The histogram output of this method is given below:

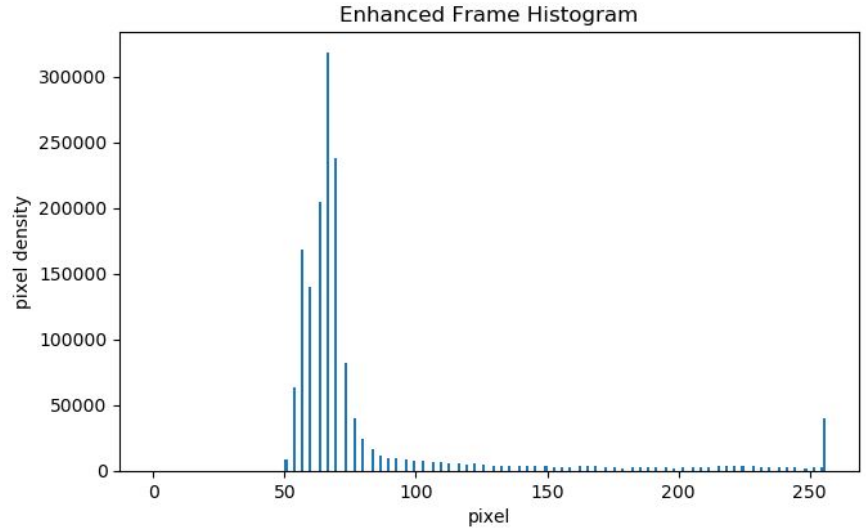

Figure 4: Enhanced Frame output

We can see that the pixel density is distributed and at 255 there is sharp rise in the pixel density which is indicative of increase in contrast of the image. The equation below clearly outlines this method:

$$NewFrame = \alpha * OldImage + \beta \tag{1}$$

1.2 Outputs Obtained

1. The outputs are snippets of one of the input frame taken from the video. The input frame considered:

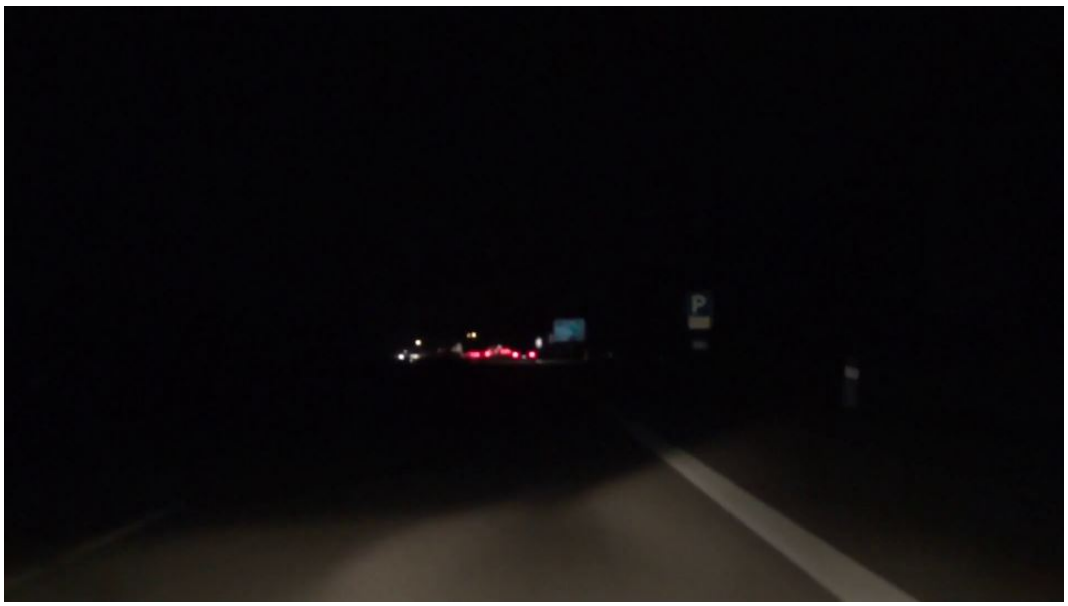

Figure 5: Input Frame

2. Histogram Equalization Output: This contains the snippet output obtained by employing histogram equalization. We can see that the contrast is improved at the same time noise is also improved.

Figure 6: Histogram Equalization output

3. Gamma Correction Output: This contains the snippet output obtained by employing Gamma correction. We can see that contrast is not improved.

Figure 7: Gamma Correction output

4. Method Employed Output: This is the snippet of the image frame from the enhanced video. We improved the contrast and decreased the noise levels.We can clearly see the lanes and road signs from the enhanced frame.

Figure 8: Enhanced frame output

1.3 Accessing output video

For accessing the full length output video please click on this link.

1.4 References:

1. Opencv documentation for improving the contrast and quality of the image.

2 Problem 2

In this problem, we are given one video and list of images and we need to provide two videos detecting lanes,

2.1 Procedure employed

2.1.1 Step1: Prepare the Input

1. For given list of images we had an additional step of first converting them into a video
2. First we used in-build Opencv function to undistort the image using the given camera parameters.
3. Then we used Gaussian filter to reduce noise in the image.
4. Then we used in-build Opencv canny edge detection function to extract the edges from the image.
5. For region of interest, we cropped the upper part of the image which is the sky and made lower part of the image as our region of interest(ROI).

2.1.2 Step2: Detect Lane Candidates - Histogram of Lane Pixels

1. First we took 4 points, 2 from each lines to compute the homography.
2. Then using in-built Opencv *warpperspective*() function, we warped the image and got the top view of the road.
3. We are only interested in lanes. So we removed unwanted color pixels by converting RGB image to HSL image. This will highlight only the lanes.
4. There were two colors with high pixel density on the HSL image which were white and yellow. We used the concept of masking to separate these two colors. By using *inrange*() function to separate the yellow and white pixels in mask1 and mask2 separately.

   Mask1 contains only the white pixels of the lane. Mask2 contains only the yellow pixels of the lane.
5. Mask1 and Mask2 are now separate anded with the image with edges(Output of Canny) using *bitwiseand*() function in opencv. The resulting two images will contain only lanes which are again ored using *bitwiseor*() function to get a single color image consisting of only lanes with edges.
6. Now we need to know the indexes of the pixel which corresponds to white and yellow pixels. For this, we converted the HSL image to gray image and obtained histogram of the image.
7. On histogram we see two peaks, which correspond to white and yellow pixels got from the lower half the image. We divided the lower half of the lower half of the image into two smaller frames. These smaller frames are divided into two categories based on their position with respect to the midpoint of the histogram. The frames on the left of the midpoint are for left (yellow) lane and right are for right (white) lane. The height of these frames depends on how much we are dividing the lower half of the image.

8. We create two lists for yellow and white lanes. We loop through these smaller frames and if any pixel is found with non-zero value we add the indexes into the corresponding yellow and white list.

9. We do this for the entire lower half of all the frames of the video.

10. Then we use $ployfit()$ function to fit a polynomial along the white and yellow lines. Then $fillpoly()$ function fills the polynomial between the white and yellow lanes.