ENPM673 Project 2

Prannoy Namala (pnamala)

April 2021

Problem 1

The problem requires us to increase the contrast of a video (of a dashcam recording), recorded at night in very dark conditions. The approach used to increase the contrast of each frame is histogram equalization. There are various ways in which histogram equalization can be done but two of the methods are explored for this problem.

Using color spaces and applying equalization on one channel

This one of the common methods to apply histogram equalization on a color image. The color space chosen here is HSV (Hue Saturation Value). The pipeline goes like this.

- Convert the selected frame to HSV by using opencv inbuilt functions.
- Separate the h,s and v channels.
- A list of count for intensities is stored from the v channel.
- Based on the histogram, calculate the cumulative distribution by adding all the values of intensities less than the current for each intensity.
- The cumulative distribution values for each pixel is then multiplied by 255(highest intensity value) and divided it by the total number of pixels in the image.
- Replace the old intensity values with the newly calculated values in the previous step. This will give us equalized v channel.
- The equalized channel is then merged with the h and s channel to create the new image.

The comparison of the applied program with other methods is shown in the figures 1 and 2.

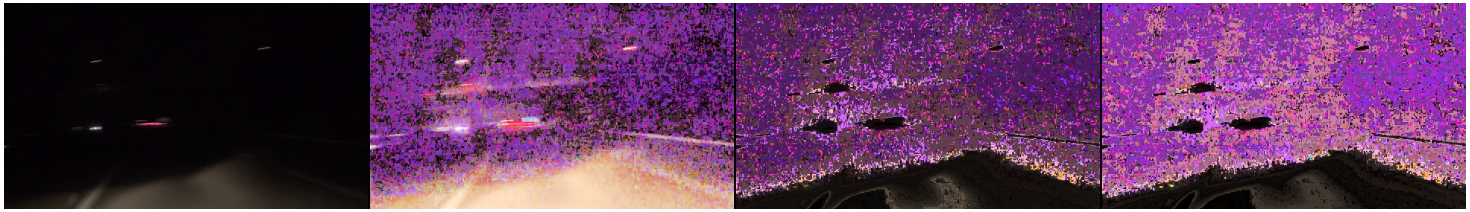

Figure 1: Comparison of original image from video to the same image equalized in different methods. From bottom, the first image is the unequalized image, the second image is the output of the inbuilt function, the third is the implementation for the project and fourth one is the implementation taken from opencv tutorials[1]

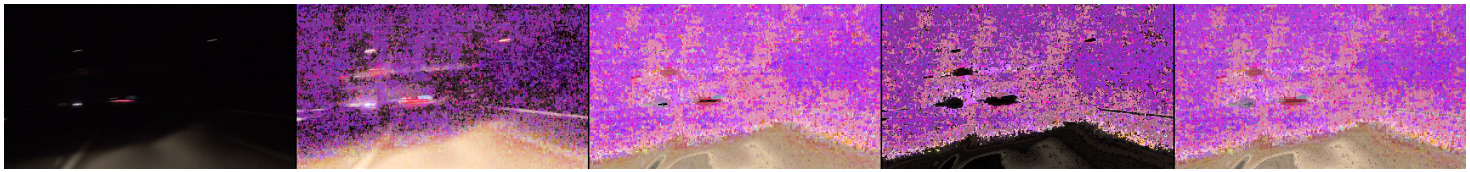

Figure 2: Comparison of original image from video to the same image equalized in different methods. From bottom, the first image is the unequalized image, the second image is the output of the inbuilt function, the third is the implementation for the project with gamma correction to aid the process, fourth one is the implementation taken from opencv tutorials[1] and the fifth picture is using CLAHE (Contrast Limited Adaptive Histogram Equalization) taken from opencv tutorials[1] with gamma correction.

Figure 3: Original image before going through equalization pipeline

Problems faced

As you can see in the image, there is a lot of noise associated with the equalized image. This is partly due to the video and partly due to twisting of colors due to change in one channel.

Equalizing a gray scale image

The pipeline for equalization remains same for a gray scale image barring the step of separating channels as there is only one channel when it comes to gray scale images. The video result is linked here. The pipeline implemented for a random image is shown in figures 3 and 4.

Problems faced

As mentioned for previous section, the noise in video was very evident while equalizing the image for the case of video. As you can see for the case of test image, the program works perfectly improving the contrast of the image.

Problem 2(a)

This problem involves detecting lanes in a given video feed. There are multiple approaches to get the required lane ad turn prediction. For this project, the

Figure 4: Processed image after going through pipeline

following steps are followed.

- **Getting the region of interest**: The first step is to extract an ROI to work on. This is achieved by cropping the image by about half along the width and taking the a triangular region which consists of only the lane and most of the other noises in the frame are eliminated.

- **Getting the bird's eye view** : The next step involves getting the Birdseye view of the ROI. This is done getting 4 points on the ROI which are approximately near the lane lines and the lines drawn from these points meet at horizon of the image. These points are used to compute homography matrix along with 4 other points which map to the chosen points in Birdseye view. The homography matrix is used to compute the Birdseye view image of a chosen size.

- **Detecting lane and direction** : Using the Birdseye view, the lane lines are then filter out using color spaces. Similar to problem 1, HSV color space is used here as well. A binary image is created using the color space where the non-zero(white)spaces are the lane line pixels. A histogram of the count of white pixels in each column is used to determine the location of lane lines on Birdseye view. Using the location, all the white pixels locations in a window of 4 columns are taken and categorized into left and right. Using these locations, the equations of lanes are found. The parabola estimate found for lanes is used to find the peak of the line. Using

Figure 5: Pipeline applied on data 1

the peak and the first point of the non-zero list, slope between these points is used to determine the direction. The last and the first point of each lane line is used to compute the lane area.

- **Laying detected area over the original image**: The final step is to show the result on the video feed. The lane area plotted is projected back to the video feed using the inverse of homography. The direction estimate is then printed on the corner of the image.

Figures 5 and 6 show implementation of pipeline on one of the image from data one and data 2 respectively.
Video 1 - here
Video 2 - here

Problems Faced

The confidence in the detected turn direction is really low. This is due to the fact that the points chosen for homography are estimates do not exactly match the four corners of the rectangle as they are supposed to for a perfect Birdseye view. Since the Birdseye view is not that perfect, the equations formed on such plane does not tend to give perfect results. Additionally, the disturbances in data 2 such as sudden blackening out under the bridge and black patch covering the white lane in some frames made it harder for recognizing the lane for some portions of the video.

References

1. Opencv tutorials - linked here

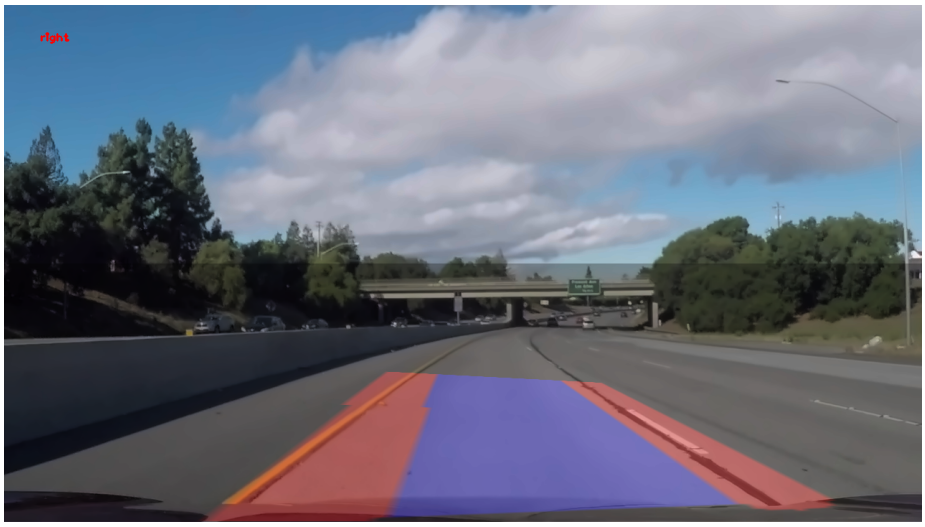

Figure 6: Pipeline applied on data 2