Project 2

Mrugesh J. Shah
117074109

A. JAMES CLARK
SCHOOL OF ENGINEERING

ENPM 673

—

PERCEPTION FOR

AUTONOMOUS ROBOTS

—

Dr. Mohammed Samer Charifa

Contents

Problem 1

The first problem is about improving the quality of the video in lowlight conditions. The goal here is to process each image frame using suggested or improved techniques to get the results that satisfy the lighting and color requirements for good features detection.

1.1 Introduction

The task here is to implement the contrast enhancement and visual improvement techniques such as Histogram equalization, Contrast adjusted histogram equalization or Gamma correction method. The current project will include Histogram equalization and Gamma correction methods for contrast correction. As well as this report will include advantages of gamma correction over histogram equalization.

1.2 Histogram Equalization

The histogram is the approximate representation of the distribution of the numerical data, and it can be used to summarize distribution of univariable data sat graphically. The major application of Histograms apart from Image Processing domain is in statistics where it is useful to for spotting deviations, verifying equal distributions, identifying data summery and much more.

Here, we have used the histogram equalization to show the distribution of frequency of each intensity value of a given image. The X-axis in our case will be all possible values for pixel intensity, while the Y-axis will be the number of pixels that has pixel intensity corresponding to point in X-axis. In our case the X-axis will range from [0, 255].

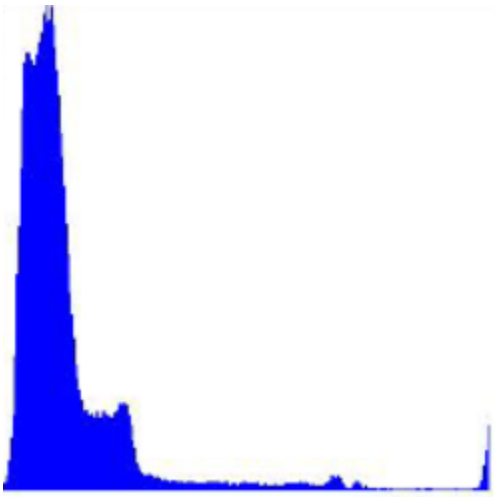


Figure 1. Histogram

Figure 2 Under exposed image

Here Figure 1. Represents the histogram of the Figure 2., which is an under exposed image of which the histogram is shown in Figure 1. Note that the image is under exposed not undersaturated or the pixel value is non-zero.

It is important for histogram equalization that the image is either under exposed or over exposed but NOT over or under saturated. As histogram equalization works by making better use of the complete color spectrum. Having mostly 0- or 255-pixel value will not show any improvements if not degradation on the image.

The idea behind histogram equalization is to modify the intensity distribution of the image. The resulting distribution of the histogram will be linear. The idea is to find the cumulative sum of the intensity of each intensity value.

The cumulative sum of the histogram is useful to get the monotonic function on which we can apply the Histogram equalization. The Figure 3. Represents the cumulative sum of the given histogram.

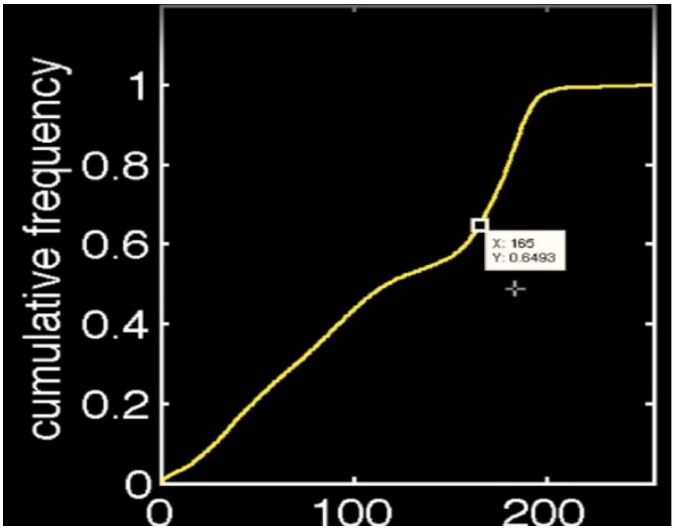


Figure 3. Cumulative sum of histogram

Now, we have to linearize this curve by following equation

$$Equalized\ intensity\ of\ pixel = 255 \times Cumilative\ frequency\ at\ that\ pixel\ intensity$$

After which we have to change all the intensity values to the equalized intensity values for each pixel in the frame. Figure 4. Represents the Histogram equalized image of the given image which was over exposed. It is important to note that the Histogram equalization will use the complete spectrum. In this case it was performed on the gray scale image, but if performed on a RGB image, we have to do same for each red, green and blue channel separately. Then combining all of them to form an image in RGB color space that is equalized.

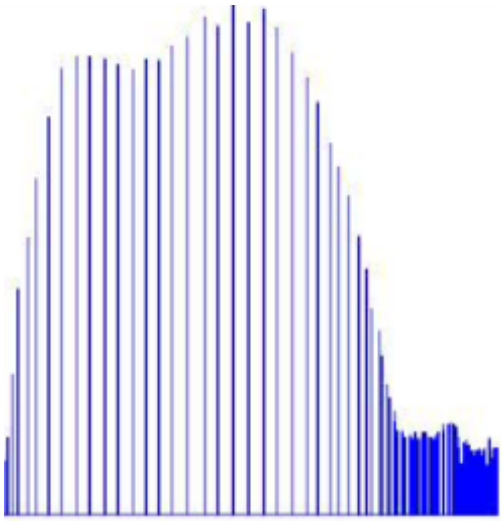


Figure 5. Equalized histogram

Figure 6. Histogram equalized Image

Figure 5. is the histogram plot of the image in Figure 6. Which was derived by assigning the pixel intensity by the given equation. Here we can see that the details which was not visible in the image earlier can be seen. The people walking under the bridge, for example, were not visible before equalization.

1.3 Problems with histogram equalization

As we can see from the algorithm of Histogram Equalization, the linearization of the cumulative sum of the frequency is causing the issue. The example image was in fact an under exposed image, but some time the output or processed image will have severe washed-out appearance as the input images dynamic range is small.

This is the case for the given frames as the data set. The implementation of Histogram equalization will give result with several pixels being washed out and appear complete white or black. Figure 7. is the output of applying the Histogram equalization on the given data set.

Figure 7. Histogram equalization on given data set

The data set, which was given, had the lower dynamic range, and thus the output looks like it has been washed out. The issue primarily lies with the data set, but the Histogram Equalization NOT being able to modify the cumulative frequency function to a nonlinear one is also an issue.

That is when the gamma correction is useful. We will understand how the gamma correction works. But first we need to know why we need to understand why we need to move from RGB color space to HSV/HSL/LAB color space.

1.4 Image format and why they matter?

The formats we dealt with till now was RGB or BGR. Both were not having separate values for luma (Light intensity) and Chroma (Color). The value of each channel represented the intensity of a color.

When we get RGB color space values of our hand, and then the palm, both will have a different R, G, and B values. But if we do same in HSV (Hue, Saturation, and Value) color space, the Hue will be same for both the cases.

Thus, for color extraction purposes, in Image Processing domain HSV format is used for color extraction or contrast adjustment and much more. The HSV format has separate elements for chroma (H and S) while the Luma or luminance is represented by the V channel.

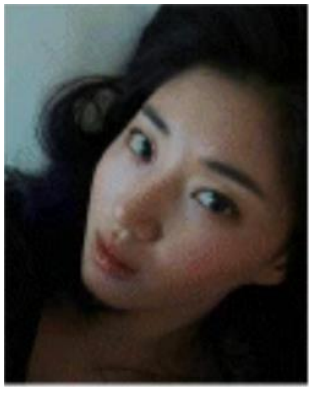
We can now just implement out algorithm on the V or light intensity only, keeping the colors of the image intact. Making the contrast adjustments look more realistic.

1.4 Gamma correction

Gamma correction overcomes the inability of the Histogram Equalization to generate nonlinear cumulative frequency graph. This will solve the issue of the low dynamic range of the input image.

(a)

(b)



(c)

Figure (a), (b) & (c) are Original Image, Histogram Equalized Image and Gamma Corrected image. We can see that the image (c) has color that is more like the original one and the lighting is also better for it.

The equation to calculate the gamma corrected pixel intensity is as follows

$$New\ intensity = \ ((Old\ intensity)/255)^{inverse\_gamma} \times 255$$

The new plot for the cumulative frequency will be as shown in Figure 8.

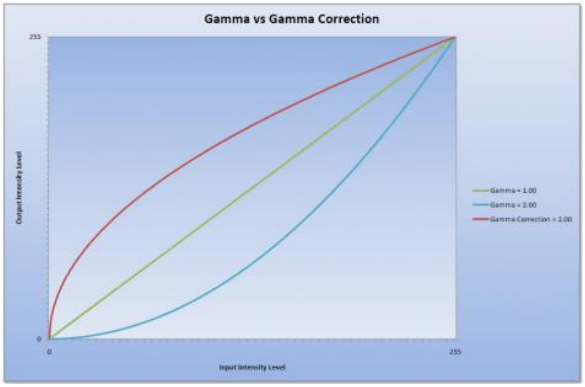


Figure 8. Gamma correction cumulative frequency histogram plot

1.5 Implementation pipeline

The first step was to get the frame from the video, and then converting it to the HSV format. Figure 9. Is the result of the conversion.

```
HSV_image = cv2.cvtColor(BGR_image, cv2.COLOR_BGR2HSV)
```

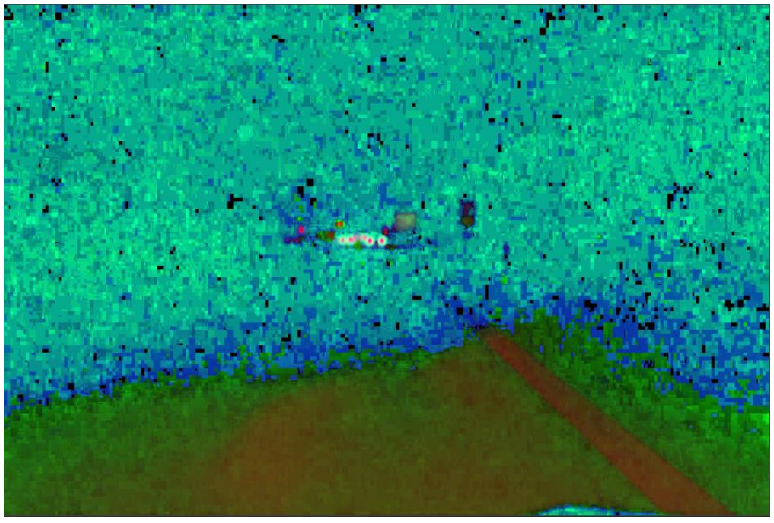


Figure 9. HSV of given data

Accessing only the V channel and applying the gamma correction to the new grayscale image with the intensity representing the value channel of the HSV image. Figure 10 represents the V channel image.

```
HSV_v = HSV_image[:, :, 2]
```

Figure 10. V channel of HSV image

Now, we will apply the gamma correction to the image with gamma value set to 1.5, note that this value of gamma is experimental and can be different for different scenarios. Figure 11 is the implementation of gamma correction on the image of Figure 10.

```
Lookup_table = np.array([((i / 255.0) ** invGamma) * 255 for i in
                         np.arange(0, 256)])

frame_gamma = cv2.LUT(HSV_v.astype(np.uint8), Lookup_table.astype(np.uint8))
```

Figure 11. Gamma corrected V channel of HSV image

Now, we have to set new intensity values to the HSV image, for that we need to change the last channel of the HSV image and then convert it to the BGR image. Figure 12. is the result,

```
                HSV[:, :, 2] = frame_gamma
BGR = cv2.cvtColor(HSV, cv2.COLOR_HSV2BGR)
```

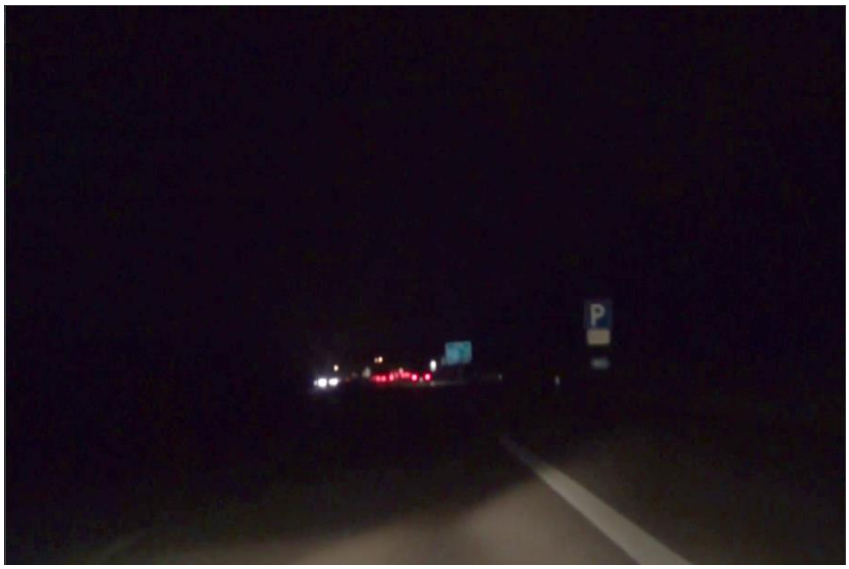


Figure 12. Final Gamma corrected output

1.6 Why it generates better results?

The gamma correction takes a nonlinear approach to generate the cumulative frequency histogram. Thus, the lack of dynamic range is compensated by the curve as shown in Figure 8. Increasing the value of Gamma will decrease the value of inverse gamma, in turn making the image darker. The calibration of gamma is based on what is the current dynamic range of the image.

If the current dynamic range is less, then increasing the correction coefficient will cause the image to appear washed out in certain regions.

Problem 2

In this problem, the task is to detect the lanes in two different data sets as well as predict the turn based on the tracked lanes. The lane detection algorithm is useful in autonomous or self-driving cars. The car will steer based on the predicted lane turn.

2.1 Introduction

In this problem, we have to first get the lane in a form that is possible to be processed to get the lane parallel, there are two suggested pipelines for this application. One uses the Histogram analysis that we learnt in previous problem and another one includes the concept of Hough lines. Both methods generate sustainable results, yet I have chosen the Histogram Analysis method for the lane detection.

2.2 Homography

The idea behind homography is to project an image from our perspective to the perspective of the camera. The homography matrix is a 3x3 matrix with [3, 3] element usually 1 representing its scale invariance.

Homography matrix is used to project an image on a plane that is not aligning with the current frame of reference either parallelly or perpendicularly. To put it into simpler words, let's say we have an image in rectangular shape, and we want it to place on a plain that extends to horizon. The shape of image will now become trapezoidal.

It is used to add the 3D effect to a 2D image as well. This matrix will transform the image to give it a perspective that matches with the scene. One of the major fields that uses Homography matrix is Augmented Reality (AR).

The use of the matrix is to convert the homogenous coordinates of the one frame to another with the help of the Homography matrix. For that, you must generate the matrix and that will require at least four points pair. The following is the homography matrix calculation method.

$$A = \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1 * x_{p1} & y_1 * x_{p1} & x_{p1} \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1 * y_{p1} & y_1 * y_{p1} & y_{p1} \\ -x_1 & -x_1 & -1 & 0 & 0 & 0 & x_2 * y_{p2} & y_2 * x_{p2} & x_{p2} \\ 0 & 0 & 0 & -x_2 & -y_2 & -1 & x_2 * y_{p2} & y_2 * y_{p2} & y_{p2} \\ -x_1 & -x_1 & -1 & 0 & 0 & 0 & x_3 * x_{p3} & y_3 * x_{p3} & x_{p3} \\ 0 & 0 & 0 & -x_3 & -y_3 & -1 & x_3 * y_{p3} & y_3 * y_{p3} & y_{p3} \\ -x_1 & -x_1 & -1 & 0 & 0 & 0 & x_4 * y_{p4} & y_4 * x_{p4} & x_{p4} \\ 0 & 0 & 0 & -x_4 & -y_4 & -1 & x_4 * y_{p4} & y_4 * y_{p4} & y_{p4} \end{bmatrix}, \quad x = \begin{bmatrix} H_{11} \\ H_{12} \\ H_{13} \\ H_{21} \\ H_{22} \\ H_{23} \\ H_{31} \\ H_{32} \\ H_{33} \end{bmatrix}$$

The homography matrix is useful for getting the bird eye view of the lanes. The corner points are taken from the undistorted image and then used as the source coordinates. The

destination coordinates are of a rectangle, in which the lanes will be parallel. The homography will give us the matrix, which can be used for getting the bird eye view by warping based on the homography matrix.

2.3 What is warping of Image?

Now that we have the homography matrix from source to destination, we can do the perspective transform of the image. The perspective transform or the warping of the image will give us the image that looks like as if seen from a different angle. The Figure 13 will give us the example of how the image warping works.

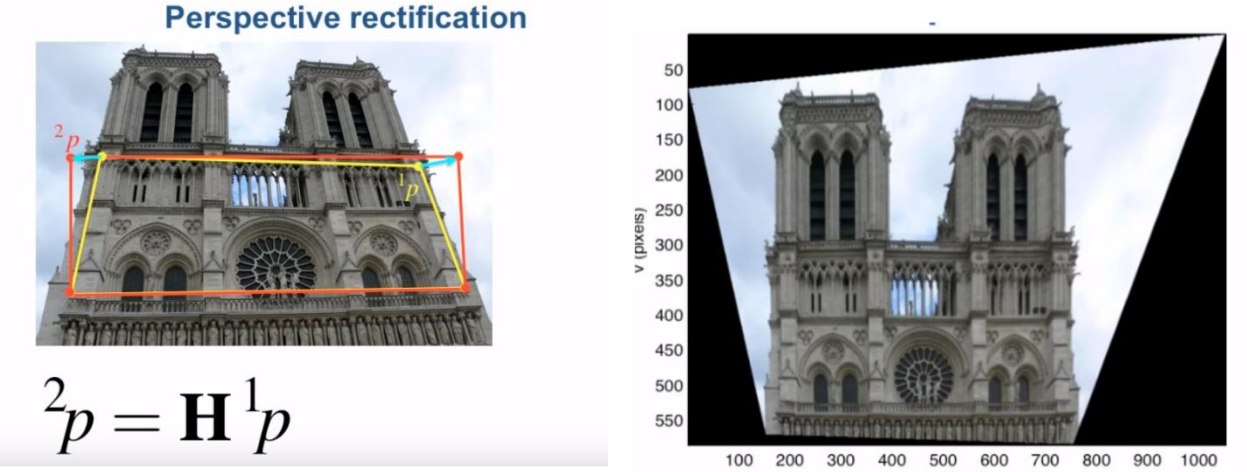


Figure 13. Perspective Transform

Here we have multiplied each pixel with the homography matrix, this will give us the new location of the pixel in warped image. It is important to note that all the calculation is happening in the Homogeneous coordinates and thus we have to understand the conversion from homogenous to cartesian coordinates. Figure 15. Explains how to do that.

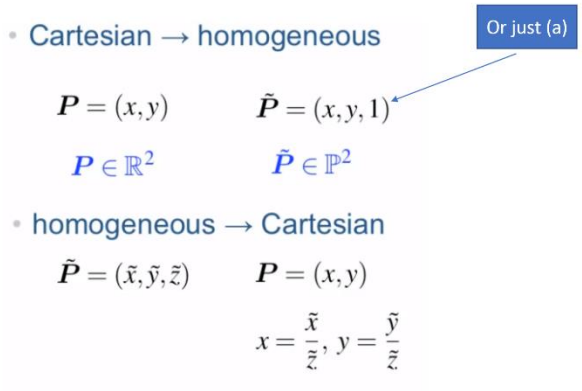


Figure 14. Conversion of coordinate systems

The use of warping is to get the bird eye view of the road to apply the turn predicting algorithm to get the orientation data. The warping of the image will take each pixel coordinates and generate new coordinates for that pixel.

2.4 Region of Interest

Region Of Interest are the samples within the data set that are identified for a specific purpose. In our implementation, we have to detect the lanes. Thus, our object of consideration will be the lanes. There is a criterion of choosing ROI, that it has to be below the horizon line.

We have selected the corner points as the region of interest experimentally. Both data set will have different ROI corner values.

Having ROI defined will lower the computing cost and false detections.

2.5 Pipeline

The first step was to undistort the image, by given camera parameters using OpenCV building functions.

```
undistorted_image = cv2.undistort(frame_given, K, dist)
```

After that, set the corner points to define the ROI for both the data set individually.

```
corners_src_1 = [(590, 275), (710, 275), (940, 515), (155, 515)]
corners_src_2 = [(610, 480), (720, 480), (960, 680), (300, 680)]
```

Now, we have to get the homography of the source points to the coordinates of the bird eye view. For that we need to get the homography matrix of the corners src X, to a new image of 500x500 dimensions. For that, we can use the function by OpenCV, the code is shown below.

```
H = cv2.findHomography(pnts_dest, pnts_src)
```

Then we need to use that H matrix to get the bird eye view. For that, we need to implement the warp perspective on the image. The OpenCV has a built in function for that,

```
warped = cv2.warpPerspective(image, H, (maxWidth, maxHeight))
```

But we built our own warp function in previous project, that can warp image flawlessly. Thus, we will continue to use that function.

```
Warped_image = warp(H, src_image, height, width)
```

The Figure 15. is the bird eye view of the lane for both the data sets. The bird eye image is still in the BGR format, note that we will change it later as HSV color space not good when it comes to visualization.

(a) (b)

Figure 15. Bird Eye View

You will notice the Figure (b) has the center portion masked out, this is because second data set had several textured details on the lane. Which were causing false detection of points and causing error by creating false peaks near the dark region after the bridge.

Now, we have to convert the image into the HSV color space. So, we can implement filters to detect the colors. The first data set is less complex as it has only one colored lane which is white. While second data set has two different colored lanes. We have to create two separate thresholds for the second data set and OR both the results. 255, 255)]]. Note, that the threshold values are chosen by examining the HSV values at the bottom of the image seen from cv2.imshow() function. The range has to be adjusted by examining the thresholded binary image. Figure 16. Represents the binary thresholded image for both data sets.

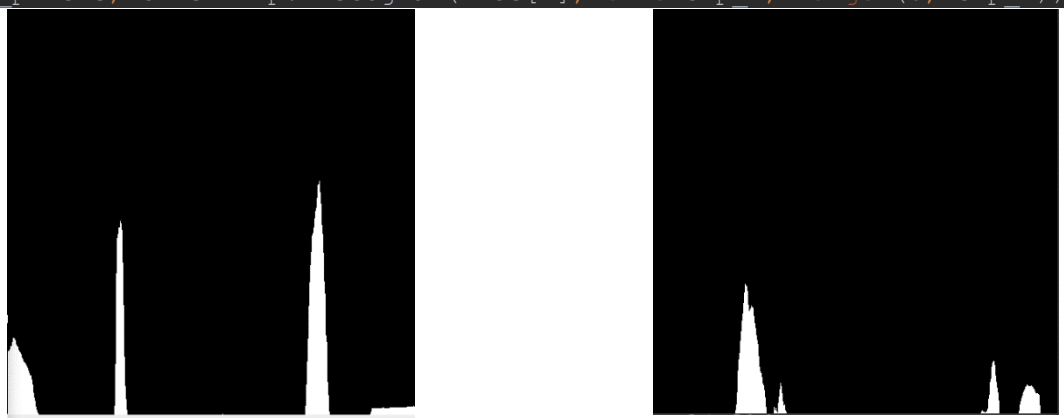
```
            hsv_img = cv2.cvtColor(given_image, cv2.COLOR_BGR2HSV)
            threshold_data_set_1 = [[(0, 0, 215), (255, 50, 255)]]
threshold_data_set_2 = [[(0, 56, 100), (255, 255, 255)], [(0, 0, 190), (255,
```

(a) (b)

Figure 16. Thresholded Images

After getting the thresholded images, we have to apply histogram analysis, but this histogram plot is a bit different from the problem 1. The X-axis in problem 1 was the intensity, but in this histogram plot, the X-axis the column index of the image and Y-axis is still the frequency, but the frequency of white pixels. The histogram plot was derived by using the following piece of code. Figure 17. Is the histogram plot using OpenCV for better visualization.

```
num_pixels, bins = np.histogram(inds[1], bins=crp_w, range=(0, crp_w))
```



(a) (b)

Figure 17. Histogram plot

As you can see that there are two peaks in the plot, we have to detect both the peak and assign them the left and right lane variables. I have also created a bound, the bound will be from the peak to certain pixel distance on both the sides. This was also taken from experiments, by analysing the width of the peaks. The peak detection was done with Continuous Wavelet Transform (CWT), but normal peak detection technique of comparing values can also be used. The only draw back was the time complexity. The CWT computes the peak quite faster than the conventional method to find the maximum value and has the built-in function in SciPy module, signal library.

```
hist_peak_all = signal.find_peaks_cwt(num_pixels, np.arange(1, 50))
```

The next step is to apply RANSAC or TLS method to generate the line from the points detected, which are lanes. I have used np.polyfit() which uses the TLS (Total Least Squared) line fitting algorithm for finding the lane coefficients. I have also used the 1st degree curve firring, which means I will only have to deal with the Slope (m) and Intercept (c). The following code is used for the TLS in NumPy.

```
Left_lane_coeff = np.polyfit(pnts_left[:, 1], pnts_left[:, 0], 1)
```

This will give us left lane variables, same can be done for the right lane. We need to create a single lane which will be the average of both of these lanes, and we will have to worry about only one slope. Here is the example of the line detected in the bird eye view, I have included only one figure, as both of them were pretty similar. So, it does not make sense to show both the images. The slope is calculated and set an threshold as well for the tilt to be considered as a turn. Figure 18 is the representation.

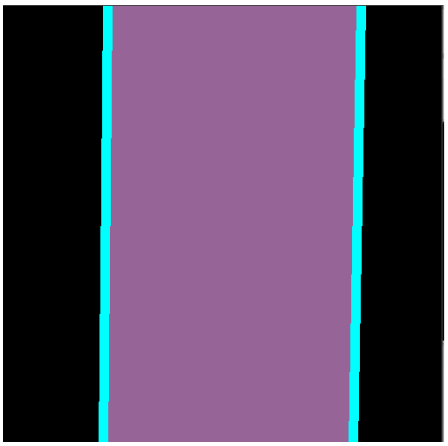


Figure 18. Lane detected in bird eye view

Now that we have the turn prediction based on the slope of the average of these two lines, but the slope of a perpendicular line is infinity, that cannot be calculated. Thus we have to rotate the image and then calculate the slope. Now, the negative slope will mean a right turn (if turned anti clockwise) and vice a versa.

Next step is to superimpose the image on the original frame. This can be done by taking the inverse of the homography matrix and applying the perspective transform. This can be done by following piece of code. Remember we have used cv2.addWeighted() for better superimposing. This function will work out way to create a semi-transparent image which will look better. Figure 19 is the final output of the frame in both the data sets.

```
                H_inv = np.linalg.inv(H)
    warped_lane = warp(H_inv, lane_image, frame.shape[0], frame.shape[1])
```

(a)

(b)
Figure 19. Final Output

This pipeline has to be implemented for each frame. There were a few cases in second data set, when there were false peaks detected on the right side. It was near and under the bridge. The patterns in the middle of the road were causing the false detection of the lane. There the masking of the middle of the lane came handy.

2.6 Problems faced

1. Selection of the color space
2. The proper thresholding for yellow and white lane was a bit complex.
3. The patterns in the middle of the road were a hassle.
4. Some shadowy region had false peaks in histogram.
5. Whether to choose histogram pipeline or Hough lines to detect the lanes. Histogram equalization was a simpler approach among both.

2.7 Hough Lines

The Hough lines are the representation of the points in a cartesian coordinate system to the Haugh plane. The use of Hough lines is to do the Hough transform. The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form.

Hough transform is most used for the detection of regular curves such as lines, circles, ellipses and many more. The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise.

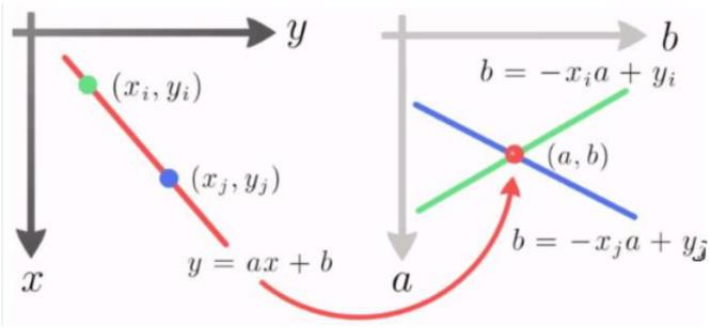


Figure 20. Haugh Lines in Cartesian coordinate system

The point is represented as the line in Hough plane, while a line is represented as a point in Hough plane. It is important to note that we can find the line by converting two points to Hough plane and then finding the intersection point in Hough plane. Which will be the line in our cartesian plane and that line is the line passing through both the points.

The problem we face with this kind of transformation is that, the slope may go to infinity for certain combination of points. Thus, we have another representation for the Hough lines in Haugh plane which includes angle (θ) and distance from origin (r) which is the polar coordinate system.

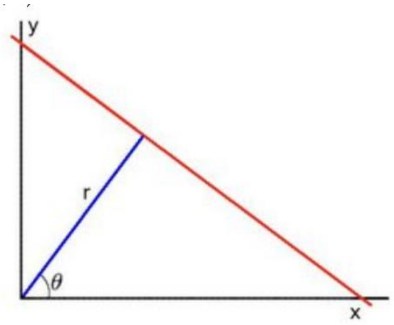


Figure 21. Haugh lines in Polar coordinate system

This follows the below line equation.

$$r = x \cos\theta + y \sin\theta$$

We can plot several different curves as shown in Figure 22. And the intersection of them will give us an edge. This is how the corner detection works with Hough lines. The process has high cost of computing, while the single pass through the complete image will give us all the edges. Thus, we only need to apply this method once for each frame.

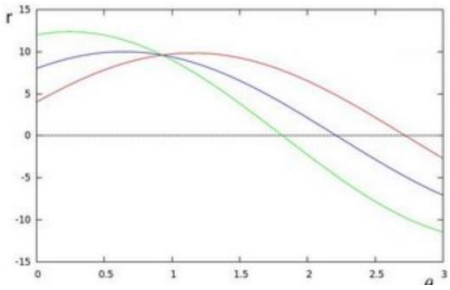


Figure 22. Multiple Hough lines in polar coordinates

2.8 Issues with generalization

There are two major issues with the generalization. The color of the lanes on the road may not be same for the complete path. The threshold values must be changed for filtering or masking out the lanes from the image.

The patterns that are in the middle of the lane are usually of the same color as the lanes. This may cause some trouble for the Histogram analysis, but we can mask that portion out as these signs are usually marked on the road signs as well.

The other issue with generalization is that sometimes in shadowy places, the lanes has different color threshold, and even with the HSV color filtering it not possible to detect.

Otherwise, on a highway at daytime, this implementation will work flawlessly.

Video Link

4.1 Drive link

https://drive.google.com/drive/folders/1Mg1YoMrj7JkNRqyNSxW1AzkB3yeetQq7?usp=sharing

4.2 YouTube link

https://www.youtube.com/playlist?list=PLXL7woBedevLcH1yT_3dWG-euK8-KXeBq

References

[1] Histogram equalization - Wikipedia
Histogram equalization - Wikipedia. (2021). Retrieved 5 April 2021, from https://en.wikipedia.org/wiki/Histogram_equaliza

[2] Histogram Equalization - Tutorialspoint
Histogram Equalization - Tutorialspoint. (2021). Retrieved 5 April 2021, from https://www.tutorialspoint.com/dip/histogra

[3] Histogram Equalization: Image Contrast Enhancement | What is Histogram Equalization?
Histogram Equalization: Image Contrast Enhancement | What is Histogram Equalization?. (2020). Retrieved 5 April 2021, from https://www.mygreatlearning.com/blog/histogram-equalization-explained/

[4] Gamma correction - Wikipedia
Gamma correction - Wikipedia. (2021). Retrieved 5 April 2021, from https://en.wikipedia.org/wiki/Gamma_correction

[5] Understanding Gamma Correction
Understanding Gamma Correction. (2021). Retrieved 5 April 2021, from https://www.cambridgeincolour.com/tutorials/gamma-correction.htm

[6] What and Why is Gamma Correction in Photo Images? (with calculator)
What and Why is Gamma Correction in Photo Images? (with calculator). (2021). Retrieved 5 April 2021, from https://www.scantips.com/lights/gamma2.html

[7] Hough transform - Wikipedia
Hough transform - Wikipedia. (2021). Retrieved 5 April 2021, from https://en.wikipedia.org/wiki/Hough_trans

[8] Line detection - Wikipedia
Line detection - Wikipedia. (2021). Retrieved 5 April 2021, from https://en.wikipedia.org/wiki/Line_detection

[9] Histogram - Wikipedia
Histogram - Wikipedia. (2021). Retrieved 5 April 2021, from https://en.wikipedia.org/wiki/Histogram

[10] OpenCV: Introduction to OpenCV-Python Tutorials
OpenCV: Introduction to OpenCV-Python Tutorials. (2021). Retrieved 15 February 2021,
from https://docs.opencv.org/master/d0/de3/tutorial_py_intro.html

[11] OpenCV: Install OpenCV-Python in Ubuntu
OpenCV: Install OpenCV-Python in Ubuntu. (2021). Retrieved 15 February 2021, from https://docs.opencv.org/master/d2/de6/tutorial_py_setup_in_ubuntu.html

[12] Homography (computer vision)
Homography (computer vision). (2021). Retrieved 15 February 2021, from https://en.wikipedia.org/wiki/Homography