**ENPM673 – Perception for Autonomous Robots – Spring 2021**

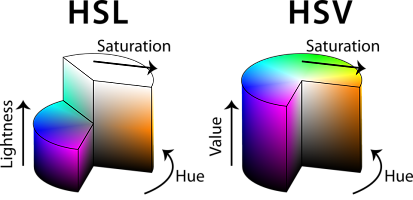
**Project 2**

**Problem1 – Histogram equalization**

The dataset given was a video which was shot by keeping a camera on the driving car at night.

The problem statement was to develop a pipeline to enhance the quality/lighting condition of the image

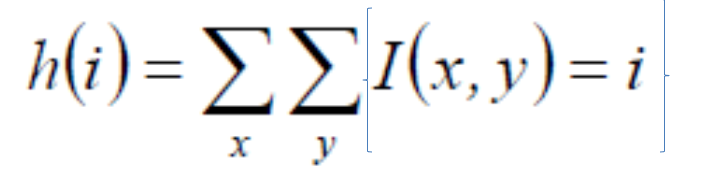
1. Calculating histogram of the given color values
   1. First calculated histogram using RGB values of the image
      1. Problems faced- Computationally expensive because need to apply histogram on 3 values
      2. Output was having mixtures of colour/was not upto the mark
   2. So converted the image to HSV then calculated histogram



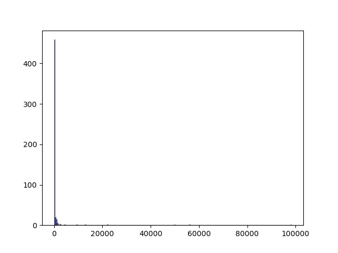
[hsl\_hsv\_models.png (413×200) (meshlogic.github.io)](https://meshlogic.github.io/posts/blender/materials/nodes-hsl-color-model/hsl_hsv_models.png)

* + 1. This decreased the computation time as histogram was calculated on only 1 value that is on V in HSV or for HSL on the L value
    2. But this added a little pink tint to the image
  1. So converted the image to YCrCb then calculated
     1. This decreased the computational time
     2. It also removed the tint as the histogram calculation was done on Y channel i.e. the luminous channel and we want to increase the lightning of the image

1. Next step is to calculate the histogram
   1. Using traditional method



* + 1. This is computationally expensive and time consuming



* 1. Using np.unique which gives unique value any found and how many times it occurred so it is basically similar calculating histogram.
     1. Advantage is it is computationally cheap and efficient and saves time

1. Step 3 is to find cumulative sum of the histogram values.
2. Step4 to find the cumulative distribution function
   1. mapping = np.maximum(0, np.round((255\*cum\_hist)/(640\*480))-1)
3. To transform/change the original image values to the new values calculated using mapping function
   1. But this gives a pixelated output and with extra 
4. To overcome above problem we use gamma correction
   1. In this we normalize the intensity channel and then raise it by the corresponding inverse gamma factor and again multiply the image by 255



Drawbacks of this image the quality of the image is not upto the mark. Can be improved by using adaptive histogram equalization

After

Before

Hist\_equal- <https://youtu.be/-qaVb3YKlyk>

**Problem 2-**

**The dataset for problem-2 has 2 different video sequences which have different camera parameters for undistorting the images and are provided separately. The first folder consists of images that you need to stitch to get the video, whereas the second folder has a video. The respective camera parameters have been included for your reference.**

**In this project we aim to do simple Lane Detection to mimic Lane Departure Warning systems used in Self Driving Cars. You are provided with two video sequences, taken from a self-driving car. Your task will be to design an algorithm to detect lanes on the road, as well as estimate the road curvature to predict car turns.**

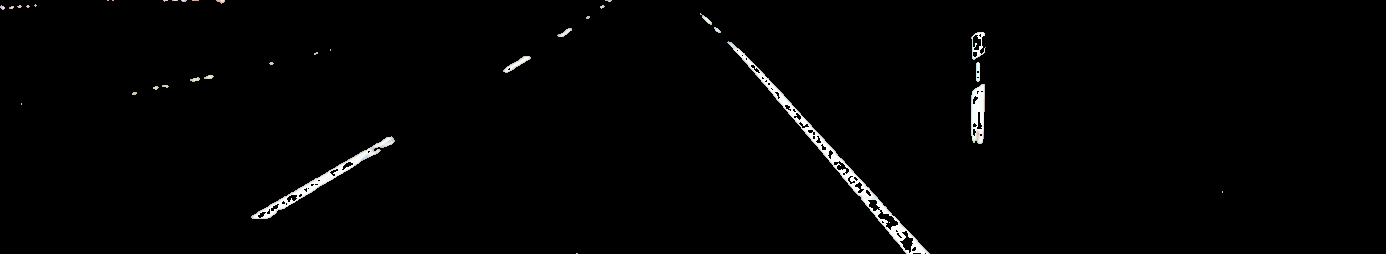
1. Step 1 - To undistort the frame of video



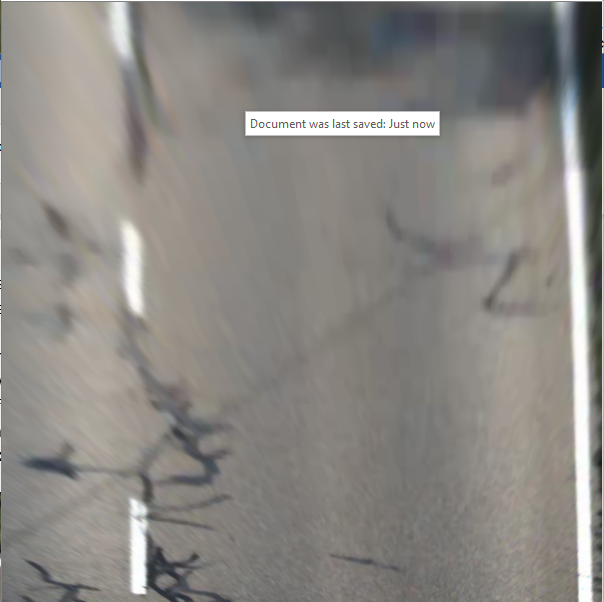
1. Step2 - To blur the image to remove noise
   1. To remove the noise from image using gaussian blur, but it smoothen outs the edges which makes it hard to extract the yellow and white line
   2. Use Median Blur which retains the edges and removes the noise which makes it easy to extract yellow and white line
   3. Step3 – Select an roi (I.e. removing the upper half of the image which contains the sky) as it wont have any influence on the lane detection but will make the implementation computationally expensive and time consuming



1. Step4- Select an image and manually extract 4 points for homography. Once pts for homography are calculated following is applied for every frame
2. Depending on the lighting condition, Adaptive histogram equalization with gamma correction is used. If this was not used then lane cannot be detected under the bridge where lighting condition is poor.(Discussed in challenges faced)
3. Use colour mask for extracting yellow and white line
   1. Using HSL as extracting yellow and white colour from it is easy than RGB

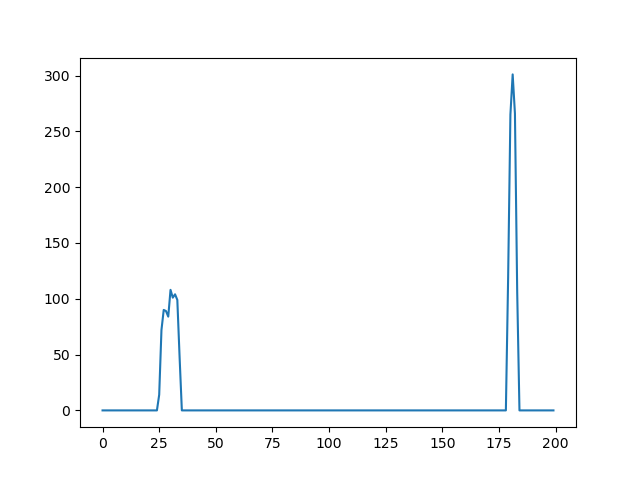


1. Step 5 calculate homography and using it wrap them to get a bird eye view

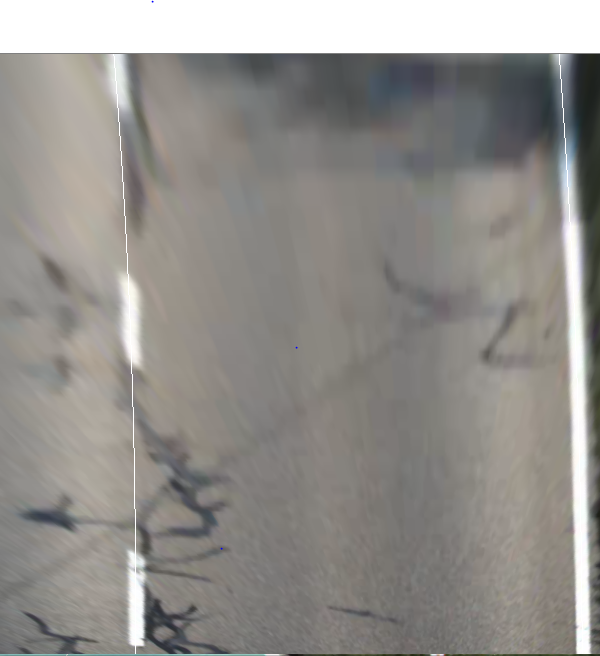
 

Once we get the mask in using warp prespective

1. We calculate histogram with respect to pixel coordinates on the mask image by converting the mask into binary we get 2 peaks which indicates the occurrence of lanes at the given pixel position



1. After getting the position of the image creating window at the position of edge/lane and searching for points of the lane.(I.e for binary image finding position where value==1)
2. After getting all index of all values use this array to fit a curve on those points . using least square(np.polyfit)



1. Fill the detected lane with colour and using the points calculated from the curve fitiing we take avg of both the lanes and take 2nd derivative/gradient and use it to calculate the turning. And inverse wrap on the road we get the final output



* Challenges Faced
  + The camera parameters in .yaml were having problem in importing so opened the value and copied the camera intrinsic parameter and undistort matrix
  + In the tunnel the lanes are not detectable due to poor condition. This was rectified using looking at the image condition and applying Adaptive histogram equalization when needed followed by gamma correlation
* Future scope
  + We could use deep learning for segmentation(SOTA) to detect the road and directly the corners of segmentation can be used with modification in the SOTA we could also get the direction od the land , or direction to take to follow the lane
  + We could use machine learning algorithm to decide when to use adaptive equalization thresholding , its kernel size and clip limit and also the gamma correlation factor
  + Using vanishing points concept we can write a function that can be used for perspective transform/homography.
* Road\_video- <https://youtu.be/XsPRxIt5g38>
* Challenge\_Video- [https://youtu.be/Ivhrjf2Ssxkt](https://youtu.be/Ivhrjf2Ssxk)

**Homography**

Homography is transformation of coordinates from one frame of coordinate to other. This is mostly from one view direction to other view. Homography is the set of parameter that help image coordinates to transform from one plane to another. It is used to transform from one plane to other plane such that those points are looked from this view. This is used to do warp perspective and for bird eye view. It is invariant to collinearity and cross ratios. Homography has 8 domain

=

Where x1,y1 are coordinate of a object from 1 plane and x1’,y1’ are same object from other plane

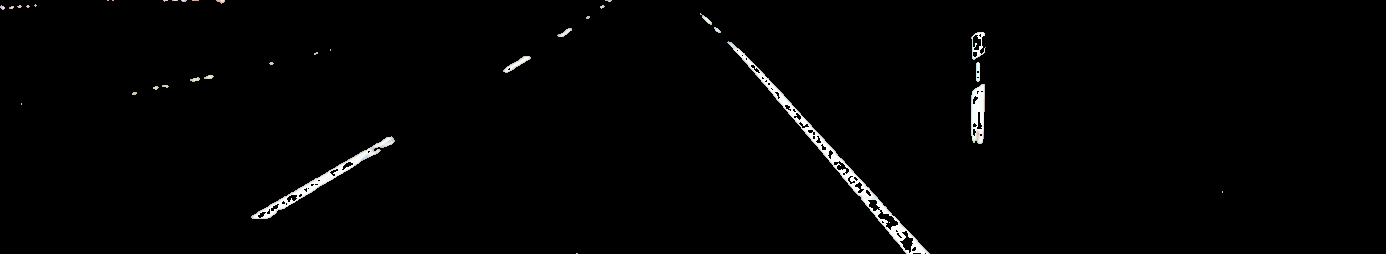


Image 1 Normal View



Image 2 – bird eye/top view

Example of homography image 1 and image 2

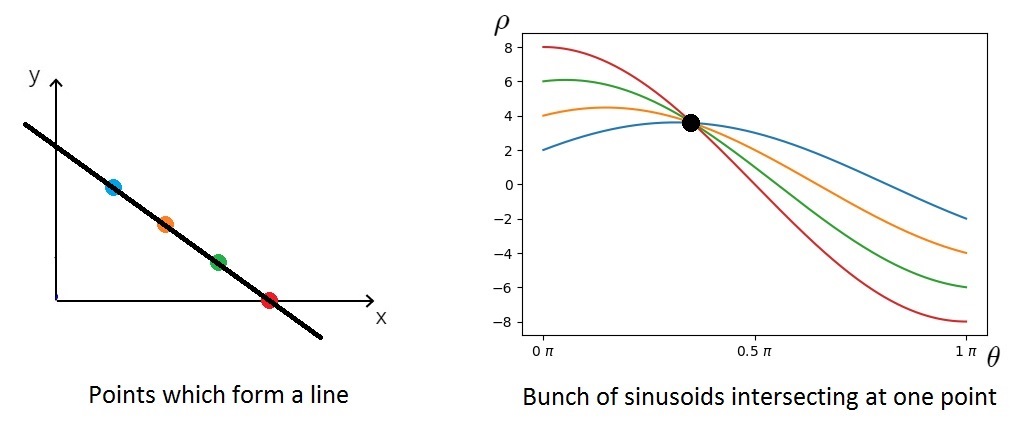
**Working of Hough lines :**

In hough transform we transform x,y coordinate to d,theta so that we can easily get the equation of line just by getting only one value of d,theta. A point in cartesian domain is sinusoidal curve in hough domain and a point in hough domain is line in cartesian domain

Intuition/key points

1. Work on voting system
2. So for each point in cartesian domain is represented by curve in d,theta domain.
3. On every point on edge we get various curves in d,theta domain all these curves intersect at same point and at this point when converted from d,theta domain to cartesian domain we get equation of line.

Actual Working



1. Converting from cartesian to x,y domain
   1. d=xcos(theta)+ysin(theta) ----(1)
2. For every point on the line the point is converted d,theta domain and in d,theta if a value occurs is updated by 1
   1. i.e. for every x,y theta is iterated from 1-180
   2. F(d,theta)+=1
3. The for other points same method is applied
4. When all points are over in an image
5. Find d, theta where f(d,theta) is large these will give you equation of line when turned back

**How likely you think your pipeline will generalize to other similar videos.**

The pipe line / the code is developed such that it worked for both the videos with very minimal changes

1. Homography point
2. Different Camera parameters
3. Video dimensions

If these values are correctly entered then the code should work with any video

As most of the parts/techniques in time line like masking and lane tracking lane detection are generalized so that it can cater to any lane ttracking. Even for bright light as well as poor lightning conditions can be handled

Video links

Road\_video- <https://youtu.be/XsPRxIt5g38> (Direction may not b visible due to image quality and background please look carefully in the top left corner)

Challenge\_Video- <https://youtu.be/Ivhrjf2Ssxkt>

Hist\_equal- <https://youtu.be/-qaVb3YKlyk>