ENPM 673

Project 2

Usnik Singh Chawla

Problem 1: Histogram equalization

Here, we aim to improve the quality of the image sequence provided above. This is a video recording of a highway during night. Most of the Computer Vision pipelines for lane detection or other self-driving tasks require good lighting conditions and color information for detecting good features. A lot of pre-processing is required in such scenarios where lighting conditions are poor.

Now, using the techniques taught in class your aim is to enhance the contrast and improve the visual appearance of the video sequence. You cannot use any in-built functions for the same. You have to use histogram equalization-based methods, both histogram equalization and adaptive histogram equalization and compare the results.

Solution:

Histogram equalization is used to normalize the values of pixels for oversaturated and undersaturated images.

In this question Histogram equalization is performed by using normal histogram equalization and adaptive histogram equalization.

Normal Histogram equalization:

In this we calculate the h vector for the entire image. In this implementation 256 bins are used. This would lead to every intensity having value in h vector. After calculating the h vector, we calculate its equivalent cumulative distributive function (cdf). After calculating the cdf vector we iterate through each pixel and get replace the intensity value for it by cdf[I[x, y]*255. This step is carried out for each channel of the colored image.

In this equalization we apply equalization to the whole image which leads to change in contrast of regions which were neither oversaturated or undersaturated. To avoid this we use adaptive histogram equalization.

Output:

Adaptive Histogram equalization:

In this we divided the image into MXN regions and apply histogram equalization for each of those areas. This gives an output with some discontinuity at the region boundaries.

This method significantly amplifies noise in the background. To avoid this we use CLACHE (contrast limited adaptive histogram equalization.)

In this the color are distorted in each of the region. This is preferably applied on Gray scale images.

Output:

Problem 2: Straight Lane Detection

In this problem we aim to do simple Lane Detection to mimic Lane Departure Warning systems used in Self Driving Cars. You are provided with a video sequence, taken from a car. Your task is to design an algorithm to detect lanes on the road, and classify them as dashed and solid lines. For classification of the line type, you have to use different colors. Use green for solid and red for dashed.

Solution:

In this problem we need to detect the lanes in which the car is travelling and separate the continuous and dashed lanes.

After loading the image, we filter out a trapezoidal area which contains the lanes in which the car is travelling. This makes this solution only valid for the given video. In order to make the solution more generic vanishing points can be used to separate out the area which contains the lane. Although this solution does work for the flipped video.

After separating out the trapezoidal area we apply binary thresholding on this area to segment the lanes.

After getting the lanes we apply canny detection followed by Hough transform to get the equation of the lines for the lanes.

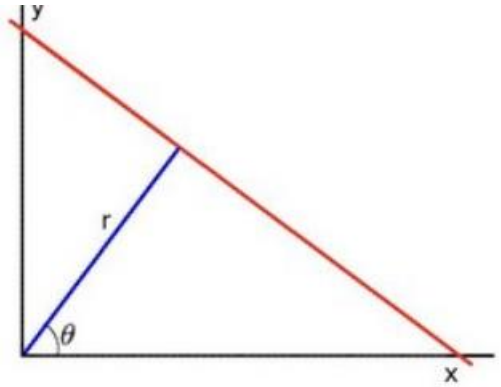
Now we separate the lines into two groups using the slope.

After we separate the lines, we calculate the Euclidean distance for the lines in each group and add them together.

The Group with the larger sum will contain the lines for continuous lane and the other will be dashed lane.

After we have identified the lines for continuous lane and dashed lane we draw the lines in the respective group with green color for continuous lane and red color for dashed lane.

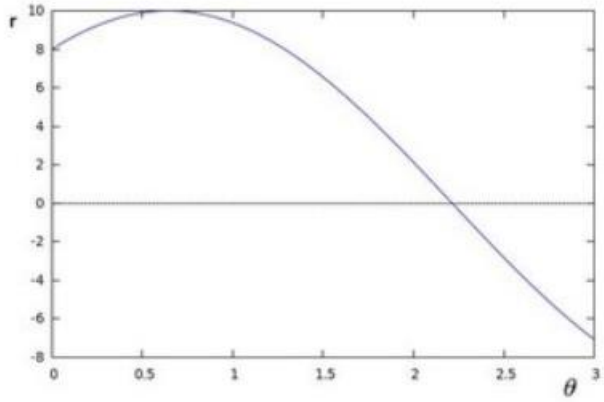
Hough Transform:



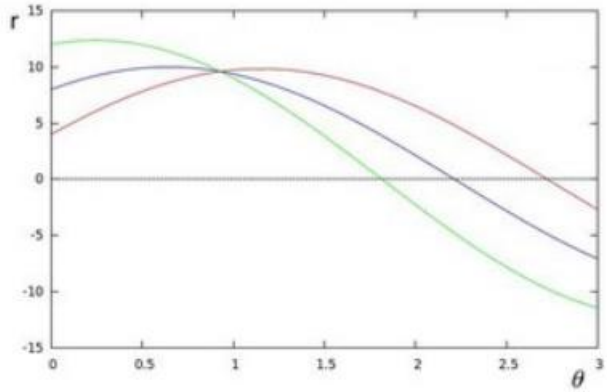
A line can be represented in:

1. Cartesian coordinate
   y=mx+b parameter(m,b)
2. Polar coordinate
   r=$(cos\theta)x + (sin\theta)y$
   parameter(r,θ)

So when we go from X,Y space to R,θ space a point x,y form a curve.



Similarly a line in X,Y space will be point in R,θ space. So when we plot curves for all the points on the line in X,Y space they will intersect at a point R, θ which will be the parameters for the line.



Output:

Problem 3: Predict Turn

In this problem, we aim to detect the curved lanes and predict the turn depending on the curvature: either left or right turn. The dataset provided has a yellow line and a while line. Your task is to design an algorithm to detect these lanes on the road, and predict the turn. Please note that for the output of the lane detection, you have to superimpose the detected lane as shown in fig.4. Also compute the radius is curvature for the lane using the obtained polynomial equation. Refer to this.

When your lane detection system loses track(cannot find lane lines), you must manage to use the past history of the lanes from the previous image frames to extrapolate the lines for the current frame.

Solution:

In this problem we detect the curved lanes and predict the turn and curvature radius.

After loading the image, we apply CLACHE on the image to remove oversaturation.

We filter out a trapezoidal area which contains the lanes in which the car is travelling. This makes this solution only valid for the given video. In order to make the solution more generic vanishing points can be used to separate out the area which contains the lane.

After separating out the Trapezoidal we warp that trapezoid to get a top view of the lane.

After getting the warp image we apply thresholding on the warp image to segment out the lanes.

Now for the first frame we detect contours for the lane.

After getting the contours we try and get points that lie inside the contours. We do this by fitting 40 lines on the image and getting the points that lie in the contours and on these lines.

After getting these points we fit second order polynomial on these points to get an equation for the lanes. Once we have them, we save them and use these for detection of lanes in other frames.

Now for frame other than the first frame we first collect all the points in the thresholded image that are not zero i.e. have a value of 255.

After getting these points we try and filter out all the points that lie close to the polynomials saved globally.

After getting these points we fit a 2nd order polynomial on these points to get the equation for lanes.

We try and check how well these predicted lanes are and accordingly update the global lane parameters so that they can be used in the next frame. If they are not good enough, we use the polynomials of the previous frame.

Once we have the polynomials for both the lane we calculate its radius of curvature using the formula

$$\text{Radius of Curvature, } R = \frac{(1 + (\frac{dy}{dx})^2)^{3/2}}{|\frac{d^2y}{dx^2}|}$$

For a specific value of x. However, in the code we calculated the polynomial for y being the independent variable so radius is calculated for specific values of y.

In code we take the value of y as 399 for a 400x400 warped image.

This gives the radius for the position of camera.

The radius here is calculated in pixels coordinates.

After we have the values for R for both the lanes, we calculate their average values.

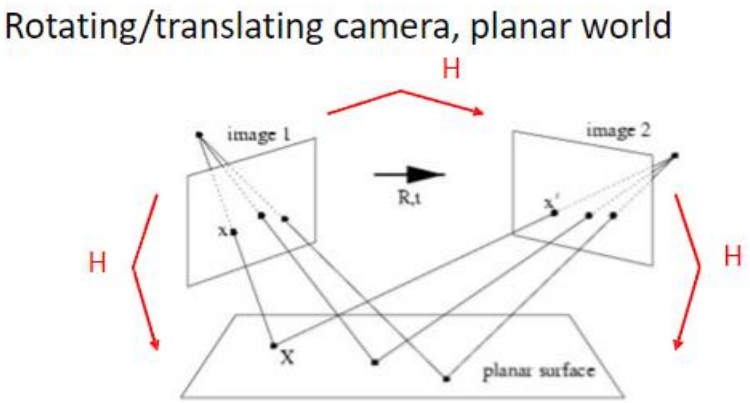
Now the direction of turns is given by direction in which the parabola equations for the lanes opens. So, we use the coefficient of the square term in the equations to give the direction of the lane.

Finally, we plot the lanes on the warped image and warp it back onto the frame to get a visualization of the detection.

Along with this we also display information regarding the direction of the turn and average radius of curvature

Homograpy:

Homograpy is basically a mapping between two planar projection of an image.



Mathematically it is given by

$$s\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Where H is the homography matrix between points in image 1 and image 2

Since we are projecting 3d points onto the two image everything we calculate will be up to some scale. So, H matrix will have 8 independent variables.

So, to get these 8 independent variables we need at least 4-point correspondence between image 1 and image 2.

Once we have these 4 points, we solve the equation A.h =0

Where A and h are

$$\underset{2n \times 9}{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_nx_n & -y'_ny_n & -y'_n \end{bmatrix}}_{\mathbf{A}} \underset{9}{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\mathbf{h}} = \underset{2n}{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\mathbf{0}}$$

h will the unit eigen vector for the for the smallest eigen value of A^TA

Output:

Links for video:

https://drive.google.com/drive/folders/16GxHtETQZAwEnL0DgnNk4vTw_vcmvai7

1. Histogram equalization-output1a
2. Adaptive Histogram equalization-output1b
3. Question2-output2
4. Question3-output3