**ENPM 673: Perception for Autonomous Robots**

**Spring 2022**

**Project 2: Histogram Equalization and Lane Detection**

**Instructor: Dr Samer Charifa**

**Teaching Assistants: Gokul Hari, Sakshi Kakdi**

**Grader: Jayes Jayashankar**

**By: Yoseph Kebede**

Problem 1: Histogram Equalization

Given a set of 25 frames, almost all in a very bright setting, it is desired that the whole image contrast is enhanced for better visual appearance such that any tasks, like ones in self driving procedures, can manipulate the image to retrieve the information they desire.

First, when looking at the gestalt of each image, it is seen that the shaded area near the trees is dark; while the road, green traffic light, white building further back, and sun light exposed trees appear bright. More importantly, there are parts of the picture that are oversaturated. For example, the crossroad is too bright which inhibits the road lanes or turn signals from being detected. Similarly, the pixels near and above the white building, though maybe not as important for the self-driving purpose (it might be where adaptive histogram equalization might make a difference in image contrast), are oversaturated. Therefore, by first using histogram equalization and later its adaptive version on all the 25 frames, an enhanced video sequence is generated with more detail.

As mentioned there are two methods used to enhance images, histogram equalization and adaptive histogram equalization, which are used in progression and finally compared to each other to see which method works best for these frames.

i) Histogram Equalization

First, histogram equalization was used to minimize the intensities with the highest number of pixels by distributing the intensities that are above a given number of pixel threshold. In order to do that, the histogram of number of pixels over intensity was plotted which enabled the algorithm to determine which intensities to cut and share with the whole image. Since the frame is a color image, to preserve the format, the input image was masked with three channels, ie Red, Blue, and Green, after which every image manipulation that followed got applied to all three. At the very end, they will be combined to a combined channel image to return to the original format. Once histogram is plotted, the threshold intensity is chosen where the intensities below or above it were divided by the total amount of pixels in the image with the cumulative frequency distributive (CFD) plot (or array of intensity to fraction) generated. Finally, by obtaining the corresponding intensity for all the intensities in the image, a new histogram was plotted (array obtained). Therefore, using this array as a conditional, every pixel got its value changed to the new amount, thus the image got equalized. After applying this pipeline for all the 25 frames, they got stitched into a video which is seen to have a different contrast than the original frames.

ii) Adaptive Histogram Equalization

When distributing the extra oversaturated pixels evenly throughout the image, while the brightness of the desired portions of the image is reduced, the dimmer regions of the image will be reduced in intensity even lower. As a result, to avoid the image contrast from being partly of appropriate intensity and partly too dark, the image will be divided into grids after which the histogram equalization pipeline is applied to each individual grid (group pixels in window). Therefore, by equalizing each grid and averaging its intensity within, the full image at the end would balance the pixels with the highest intensities in their respective windows.

Since the 25 frames have multiple oversaturated limits within them, adaptive histogram equalization method might exacerbate noise as each window might have different intensities which can not be averaged over entire frame. So, contrast limiting will be beneficial. That is, by clipping the pixel intensities with higher counts than the other intensities to a specified value, followed by averaging the count to the rest of the image, adaptive histogram method can then be followed to produce a frame with an enhanced contrast. Finally, this can be applied to all frames thereby recreating a video sequence with an improved quality.

Problem 2: Straight Lane Detection

In order to detect lanes from video of a road a car is driving through, the following pipeline is followed:

* Read frame from video
* Convert image into grayscale
* Apply Gaussian blur to avoid noise
* Apply monochromatic masking to only detect lanes
* Find homography of image to upright plane
* Warp image using homography to upright plane
* Rotate warped image to orient top view
* Use hough transforms to distinguish solid from dashed lines
* Draw green over solid lines
* Draw red over dashed lines
* Repeat for all frames
* Stitch frames into video

As indicated above, the video is read as an image frame and then converted into grayscale after which it’s filtered by masking only the critical region of the frame with Gaussian blur, converted to monochrome masking, and the interested area highlighted with the rest excluded to reduce noise. Later, the image containing only white pixels from the lanes gets the coordinates of the white pixels recorded which were used as inputs to find the homography.

Homography is an image processing method where an image is transformed from one perspective into a more easily discernable aspect (here a vertical plane image). As a result, key factors such as distance between points, line curvature, or any other shape can be discerned as if the reader is looking at the environment either face forward or top down as in birds eye view, in the direction the computation is desired. Thus, since the image frames from the video do not show the distance between the lanes thus making it difficult to compute the curvature of the lanes, homography was applied to transform the image as if it was seen top down.

Then, using this homography the image of the road was warped into a vertical plane. Thus, all the computations involving identifying the lanes, computing the curved regression lines, and drawing the curved lines on the lanes are all done on this vertical image.

HoughLines were used here to detect the lanes in this warped image. This feature works by computing the Hough Transform of the lines in the image. Therefore, by keeping a score of the curves in the radius to angle plot that cross a given point in the Hough plane for a given number of times (mentioned as threshold in the opencv function), curves in the image are determined if they are lines or not. As a result, all the lines (with given minimum length as a reference in the opencv function) will have the beginning and end points stored in an array. Thus, this is what was used to plot the lines in the warped image.

Finally, inverse warping was applied with markings appended to the original image yielding an image with lane markings. This process is repeated for all the frames, with the video created by stitching them all at the end.

This pipeline is seen to perform better as it does a good job in filtering out noise, detecting lanes (both dashed and solid), and inverse warping markings into the desired region. Furthermore, for frames with fewer number of detectable lane markings to work with, information from previous frames is used to continue proper drawings on lane. As an example, below is a figure showing the transformation of one frame using the pipeline.

Problem 3: Predict Turn

To detect lanes with curvature from a video feed and decide if turn is in a left or right direction:

* Read frame from video
* Convert image into grayscale
* Apply Gaussian blur to avoid noise
* Apply monochromatic masking to only detect lanes
* Find homography of image to upright plan
* Warp image using homography to upright plane
* Rotate warped image to orient top view
* Use quadratic regression to draw curve lanes
* Determine radius of curvature from curve
* Declare a turn or straight path based on radius
* Draw curve lines over lanes
* Inverse warp / project drawings onto original frame
* Repeat for all frames
* Stitch frames into videos

Here, like the previous problem, image frame read from video is processed (filtered, masked, homography computed, and warped into vertical plane). Then, quadratic regression curve was computed for each lane (i.e. left and right side) with the resulting points drawn over the lanes. Furthermore, using the computed quadratic equation, the first and second derivates were obtained which were plugged into the radius of curvature equation:

R = (1+(dy/dx)2)3/2

------------------

|d2y/dx2|

Once radius is obtained, the curvature of the curve (k) is computed. However, since the radius is in the range of hundreds to small thousands, the curvature will be too small to discern if the lanes are straight or curved. As a result, the actual radius is used to predict if there is a turn or not that’s approaching the vehicle.

Finally, same as problem 2, marked image was inversely warped with the resultant appended to original image. Additionally, a text was written on the final image which declared if the drive is “Predicting a Turn” or going on a “Straight Path”. This process is implemented on every frame of the video, after which all frames are stitched together into a new video.

This pipeline implements homography exactly like the previous problem; but does not use HoughLines as it produced more noise that were difficult to manage and could not cover the blank space between the dashed lines. Instead, the quadratic regression curve was used to connect the lanes to plot a continuous line in the frame. However, in hindsight, though this pipeline captures most lanes perfectly, it can be improved to work best for all type of scenarios. This is because, for couple of the frames, although image frames with robust contrast were reused when encountering images with low lighting conditions, for the high lighting exposed images the color markings were distorted after inverse warping which is attributed to the lighting changes (i.e. road in between the lanes being exposed to increased amount of sunlight) that affected the masking process. Since the algorithm, after reaching the monochromatic phase, didn’t have any means to discern if a white pixel was a lane or a random pixel in the middle of the lane, the unintended points were extrapolated and used to draw lane markings. In the future, an improvement could be made to better manage high exposure by adding features like find contours and Canny so that only the lanes alone are deduced. Nonetheless, the algorithm does a good job in achieving all the desired objectives to identify in predicting a turn from a given image of a vehicle in marked roads. Below are illustrations demonstrating the pipeline for a given frame.