Dani Lerner

ENPM673

Project 2

## Problem 1: Night Driving

In this problem, we have an under exposed video feed from a person driving at night. The goal is to use histogram equalization to widen the dynamic range of the image. The process to create histogram equalization is relatively simple. First, I created 1x256 list to store the count for each grayscale intensity level. Then I built the histogram by looping through each pixel and adding it into its corresponding bin. In the next step, I created the CDF with the equation:

)

Lastly, I replaced every pixel by its intensity’s corresponding value of C.

The gamma in the above equation helps provide a better dynamic range through a process called gamma correction. As we increase gamma, fractions close to 0 further decrease towards 0 while values closer to one barely change. This provides us with better equalization and a less pixelated, clearer image. For this assignment, I used a gamma value of 10 since it provided the best results. See the images below.

A picture containing text, night sky

Description automatically generated

Figure : Top left: original grayscale image. Bottom left: gamma = 1 (i.e. no gamma correction). Top right: gamma = 2.6. Bottom right: gamma = 10

## Problem 2: Lane Detection and Turn Estimation

In this problem, I had two videos of cars driving on a road and was tasked with detecting the lanes and turns in the road. One video was provided as a series of images, from which I had to generate the video before using. To perform lane detection, I executed the following pipeline for each video.

The first step was to preprocess the image. With a list containing the distortion coefficients, I used the cv2 functions “cv2.getOptimalCameraMatrix” then “cv2.undistort” to undistort the image and remove the barrel distortion effects. Then, I created a script that allowed me to traverse a red dot through any frame of the video and pick four corners in the image. With these manually selected corners along with 4 rectangular corners on a blank image, I could compute the homography between the camera and an apparent birds-eye view.

Then, I had to take each frame in the undistorted birds-eye view and detect and estimate the lane trajectories. To do this, I first converted the frame to HSV. Then, I thresholded the frame to find any white or yellow pixels. Unfortunately, there was a significant amount of noise in the image due to other cars and objects. However, since the road was in a birds-eye view, significant counts of pixels along a vertical line likely corresponded to lanes. Therefore, I created a histogram along the x-axis to count the number of lane candidates in each vertical line in the image. Then I filtered out any vertical line without a significant number of pixel candidates. Next, I split up filtered pixels on the left side of the image and filtered pixels on the right. With the pixel candidates split into two sides, I generated the X matrices with the “sklearn.preprocessing.PolynomialFeatures” and ran them through the “sklearn.linear\_model.RANSACRegressor” to estimate the lane trajectories. To filter out noisy frames, I calculated the approximate distance between the two lanes and filtered large the distances with a wide buffer. If the distance was far greater than the threshold, it must not have detected the lanes correctly. Therefore, it should not display the results. I then averaged the x values of the left-side and right-side predictions for every y-coordinate in the image to obtain the center prediction. Then I used the inverse of the homography matrix to back propagate the center prediction arrows and the polygon lane estimation back onto the original image. I also thresholded the change in x-values between y-val = 0 and and y-val = h/2 to predict whether the car is currently turning. The results and some intermediate steps can be seen below.

A picture containing tree, outdoor, sky, car mirror

Description automatically generated

Figure : Top: Original frame. Mid: Undistorted frame. Bottom: Processed frame

A picture containing text, different

Description automatically generated

Figure : Top left: original post-homography frame. Top right: Thresholded frame. Bottom left: Filtered lane candidates with parabolic estimation. Bottom right: Final frame