ENPM 673: Perception for Autonomous Robots **Released: Feb-26.**

Report

Name: Arpit Aggarwal and Shantam Bajpai

Problem 1 - Improving quality of video

The goal here was to improve the visual appearance of the video sequence. We followed the following steps to improve the appearance of each frame of the video:

1. First, the frame contrast was improved with "cv2.addWeighted()" function.
2. Next, gamma correction method with gamma value of 2 was used. First each frame pixel was converted between 0 and 1 and then its square root was taken. Then, the frame pixel was converted between 0 to 255 values.

Figure 1: Improved appearance of the frame

Video Output for Problem 1

The following is the video output link for Problem 1:
https://drive.google.com/file/d/1aCt2UNMoX8dIMGyb0vuH6fzhG2LCAaQZ/view?usp=sharing

Problem 2 - Lane Detection

For lane detection problem, we followed the following steps:

Pre-processing of the frame

Given the camera and distortion parameters, the frame was pre-processed. First, the frame was undistorted using the "cv2.undistort()" function. Then, average blurring was applied using "cv2.blur()" function. The image was then converted to HLS color space and conditions were applied on the white and yellow masks. The binary image was returned as the pre-processed output as shown below:

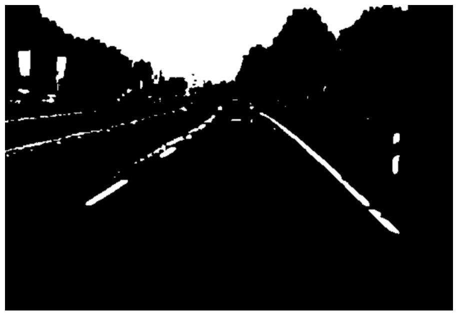

Figure 2: Binary output of the frame

Calculating homography matrix and obtaining the warped image

The next step was finding the homography of the given image to obtain the birds eye view of the lane for further analysis. The source points for "data1" were selected as:

"np.float32([[586, 293], [720, 293], [867, 507], [255, 507]])"

and for "data2" were selected as:

"np.float32([[572, 510], [789, 510], [1092, 715], [282, 715]])".

The destination points were selected as: "np.float32([[200, 0], [image.shape[1] - 200, 0], [image.shape[1] - 200, image.shape[0]], [200, image.shape[0]]])". Using this, the homography matrix was calculated. Then, "cv2.warpPerspective()" was used to obtain the birds-eye view of the image. The birds-eye view of the lane is shown in Figure 3.

Histogram of Lane pixels

Given the warped image, now the histogram of lane pixels was formed. As the lanes occur along the y-axis, the histogram was taken along the y-axis. Next step was finding the regions of highest pixel counts. There were two regions obtained corresponding to the left lane and right lane respectively. Using these two regions, the coordinates for left lane and right lane were found. These coordinates were used for finding the polynomial fit for left lane and right lane. The detected lanes in the image can be seen below:

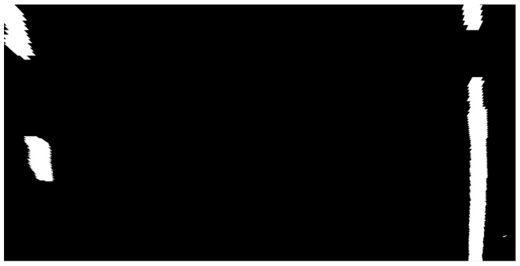

Figure 3: Birds-eye view of the lane

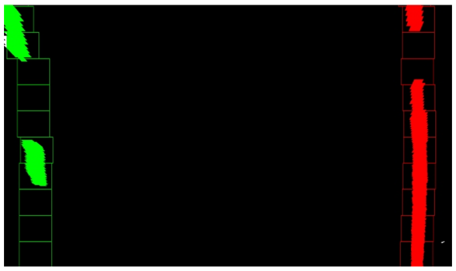

Figure 4: Detected Left and Right lanes

Radius of Curvature and Turn Prediction

Given the left lane and right lane polynomial fit, the next step was finding the radius of curvature and turn prediction. The polynomial fits for left lane and right lane were used to predict this information. The warped image was unwarped to get the detected lane back onto the original frame as shown below:

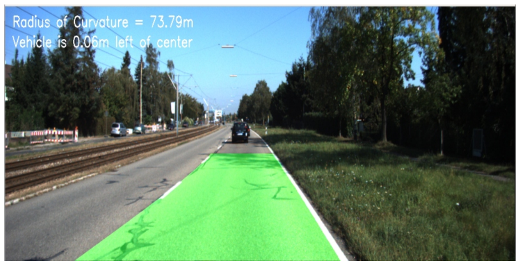

Figure 5: Detected lane on the original frame

Video Output for Problem 2 data1

The following is the video output link for Problem 2 data1:
https://drive.google.com/drive/folders/1aBmsSfCHvY2sorgT23-do3ikSuc9b-kj?usp=sharing

Video Output for Problem 2 data2

The following is the video output link for Problem 2 data2:
https://drive.google.com/file/d/16Hlf35JO3nweNTa3-1YbducV932HSi0E/view?usp=sharing

Interesting problems encountered

The following were some interesting problems we encountered:

1. In the challenge video, we faced difficulty in detecting lanes due to brightness issue in some frames.
2. We used HSL color space for finding the binary output of lanes. However, the hyperparameters for the white mask and yellow mask required a lot of tuning to work for data1 and data2.
3. In general, there were a lot of hyperparameters to tune, from preprocessing to the lane fitting step.
4. The radius of curvature is not accurate at all frames due to noise in lane detection process.

References

[1] https://www.pyimagesearch.com/2016/03/21/ordering-coordinates-clockwise-with-python-and-opencv/
[2] https://www.pyimagesearch.com/2014/05/05/building-pokedex-python-opencv-perspective-warping-step-5-6/
[3] https://www.learnopencv.com/image-alignment-feature-based-using-opencv-c-python/
[4] https://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html