Name: Chun-Yen Yeh (ID: cyyeh4), Huey-Chii Liang (ID: hcliang2)

Course: ECE484 - Principles of Safe Autonomy

Date: September 22, 2023

ECE484 - MP1 Group Assignment

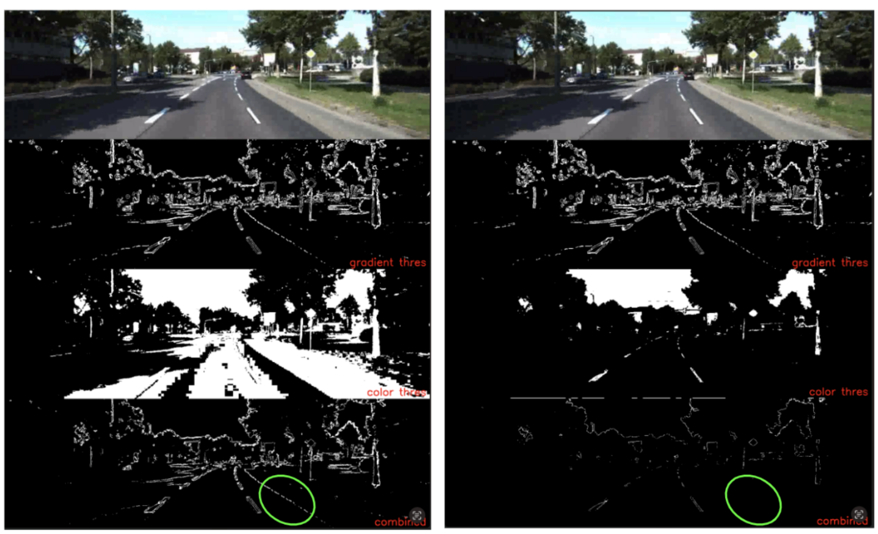
---

code link:
https://drive.google.com/file/d/1ODXTKzibaEESjn5XH_cVcAv2UycxAojM/view?usp=sharing

Problem 5 (15 points) What are some interesting design choices you considered and executed in creating different functions in your lane detection module? E.g. which color spaces did you use? How did you determine the source points and destination points for perspective transform?

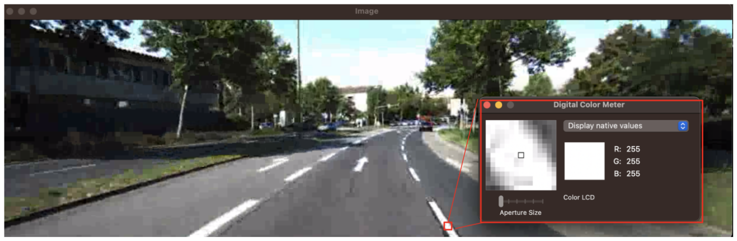
Design 1: color space

HSL is used in our MP. It allows us to filter out lanes using saturation and lightness channels. For lightness, it is trivial to detect white lanes. As for saturation, it is crucial for filtering out false positive edges along pavement junctions as the following figure shows.



w/o saturation thres **w/ saturation thres**

Note that we did not filter out pixels with saturation of 0 value. Since the saturation value equals 0 if the value of three RGB channels are the same as the following figure shows:



RGB: (255, 255, 255)
HLS: (0, 255, 0)

Design 2: src/dst points determination choosing for perspective transform

Points is determined by these two considerations:

1. Make pixels of each lane lie in the same x position after transform as the pixel histogram splits the image along the x-axis.
2. Use normal and straight ahead lanes as the base image to align. Otherwise, the transformation matrix will be screwed.

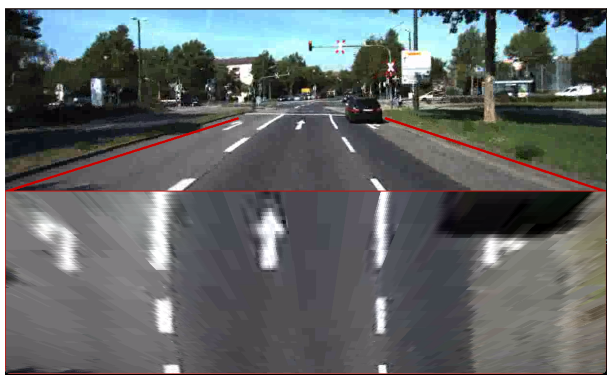
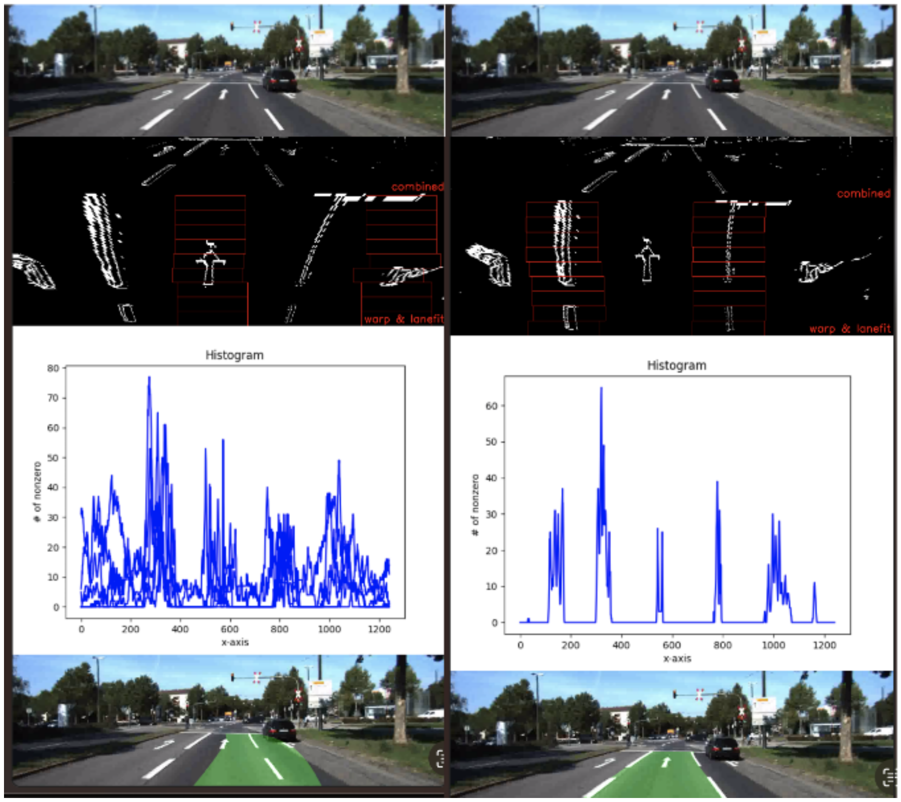


Figure 1: our base image and its transformation

Noise can be reduced if we choose the proper transformation points according to the histogram as the following figure shows.



Design 3: dilation

Sometimes gradient and color filtering results do not align well. It is because for the gradient filtering result, responsive pixels lie along the edge of lane patches. On the other hand, for the color filtering result, responsive pixels aggregate near the center of lane patches. The misalign situation is illustrated as the following:

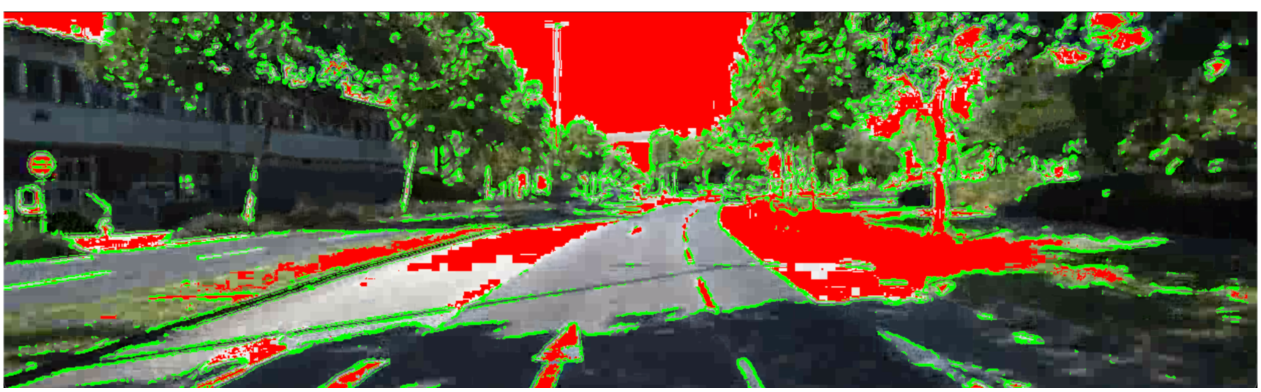
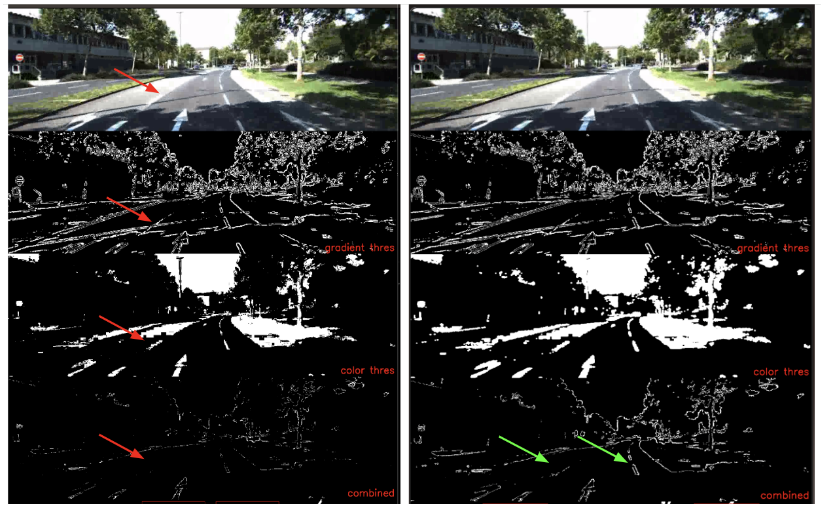


Figure: green pixels represent edge responses. Red pixels represent color responses.

As the following figure shows, to the area that is pointed by a red arrow, the lane has responses in both gradient and color filtering results. However, the response disappears in the combined result. With dilation operation, lanes once disappear are detected again.



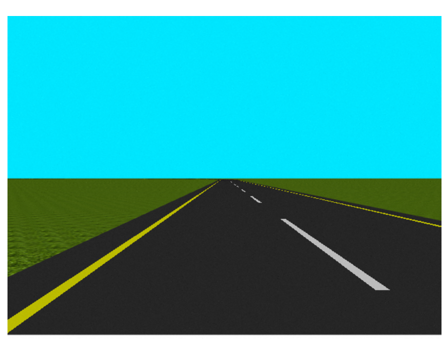
w/o dilating color filtering result

w/ dilating color filtering result

Problem 6 (25 points) In order to detect the lanes in both scenarios, you will need to modify some parameters. Please list and compare all parameters you have modified and explain why altering them is helpful?

1. Color threshold parameters
   a. For Rosbag:
      i. S channel condition: (S == 0) & (80 < S < 255)
      ii. L channel condition: (100 < L < 255)
      iii. Binary output = (S & L)
   b. For GEM car:
      i. S channel condition: (100 < S < 255)
      ii. L channel condition: (150 < L < 200)
      iii. H channel condition: (0 < H < 35)
      iv. Binary output = (S | L) & H
   c. Explanation:

      The color of the actual road scene has great uncertainty, such as the tree shadow, other cars, snow, etc., which makes it impossible for us to choose the best color threshold. However, in the Gazebo simulation, the colors of the road scene are simple (as shown in the figure below). Therefore, we can adjust color thresholds to easily filter out noise.

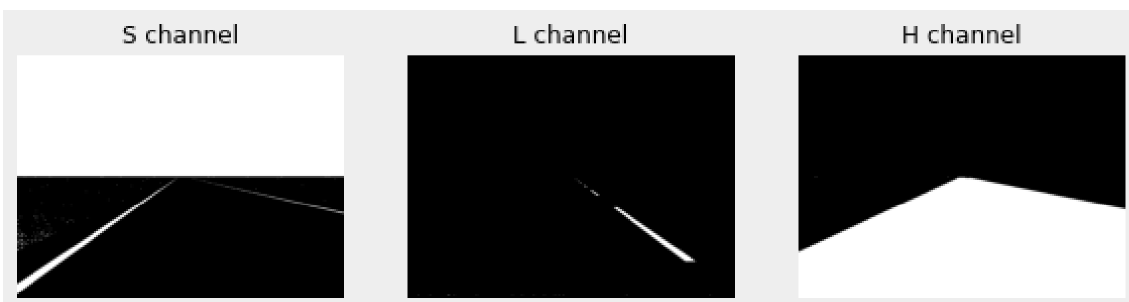


The raw image has five parts, including "blue sky", "green grass", "yellow solid line", "white dotted line", and "black road". After converting to HLS, we split the image to three channels, and we roughly recorded the values of the different parts, as shown below.

	Sky	Grass	Solid line	Dotted line	Road
S channel	255	150~220	255	0	0
L channel	128	40~80	90	190	40
H channel	90	40	30	0	0

According to the table, the S channel filters out the dotted line, the L channel makes the dotted line more obvious, and the H channel differentiates what is outside the road (the sky and grass) and what is inside the road (including lines).

Therefore, we set the S threshold to be larger than 225 to filter out grass; we choose the L threshold to be between 150 to 200 to retain the dotted line; and we select the H threshold to be smaller than 35 to leave only the area within the road. The results are shown below.



To combine these three channels and retain correct information (only the solid line and the dotted line), we set output equals to H & (S | L). It means that in the H channel, we only consider the area within the road, and we want both the solid line in the S channel and the dotted line in the L channel. The final output of the function color_thresh is shown below.

2. Src/dst points for perspective transform.
   a. For Rosbag:
      i. Source points: (image size = [1242, 375])

Upper left = [481, 224]	Upper right = [786, 224]
Lower left = [0, 375]	Lower right = [1242, 375]

      ii. Destination points: (image size = [1242, 375])

Upper left = [0, 0]	Upper right = [1242, 0]
Lower left = [0, 375]	Lower right = [1242, 375]

   b. For GEM car:
      i. Source points: (image size = [640, 480])

Upper left = [640*0.40, 480*0.55]	Upper right = [640*0.60, 480*0.55]
Lower left = [640*0.00, 480*0.75]	Lower right = [640*1.00, 480*0.75]

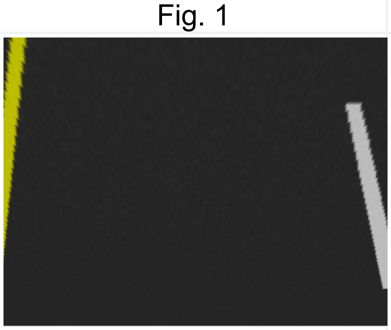
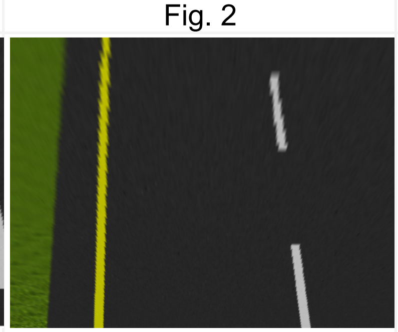
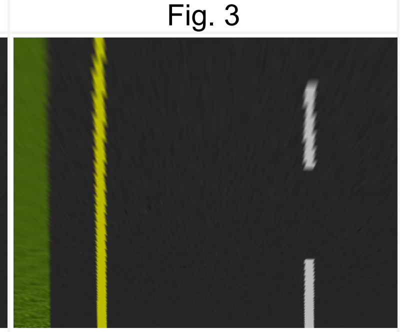
      ii. Destination points: (image size = [640, 480])

Upper left = [0, 0]	Upper right = [640, 0]
Lower left = [0, 480]	Lower right = [640, 480]

   c. Explanation:

In rosbag, we find the specific source points by looking directly at the picture. However, the picture size is different from the size of the Gazebo test image, so we reselect the source points based on the Gazebo test image.

First, we convert the original source points into proportional expression, then apply them to the Gazebo test image, as shown in Figure 1 below. Due to the perspective difference between the two scenarios, the bird-eye view is too close to the road. Thus, we reselect the source points upwards and reduce the area composed of these points. Fig. 2 shows the result, we can see that the scale of the bird-eye view is more suitable. In order to make the two lines parallel, we reduce the distance between the upper left point and the upper right source point, resulting in the final bird-eye view as shown in Fig. 3.

Fig. 1	Fig. 2	Fig. 3
		
Upper left = [width*0.38, height*0.60] Lower left = [width*0.00, height*1.00] Lower right = [width*1.00, height*1.00] Upper right = [width*0.64, height*0.60]	Upper left = [width*0.38, height*0.55] Lower left = [width*0.00, height*0.75] Lower right = [width*1.00, height*0.75] Upper right = [width*0.64, height*0.55]	Upper left = [width*0.40, height*0.55] Lower left = [width*0.00, height*0.75] Lower right = [width*1.00, height*0.75] Upper right = [width*0.60, height*0.55]

Problem 7 (30 points) Record 2 short videos of Rviz window and Gazebo to show that your code works for both scenarios. You can either use screen recording software or smart phone to record.

- 0011 video
- 0056 video
- 0830 video
- Gazebo video

Problem 8 (20 points) One of the provided rosbags (0484_sync.bag) is recorded in snowfall condition. Your lane detector might encounter difficulties when trying to fit the lane in this specific case. If your lane detector works, please report what techniques you used to accomplish that. If not, try to find the possible reasons and explain them. (Note that you will not be evaluated by whether your model fits the lane in this case; instead we will evaluate based on the reasoning you provide.)

Our algorithm fails to fit lanes in snowfall condition as shown in the following figure. This difficulty results from our algorithm's reliance on the lightness and saturation of lanes for detection. However, these two properties of lanes in snowfall conditions are significantly different from those in normal circumstances.