Project 2 Report

ENPM673 SPRING 2021

Divyam Garg

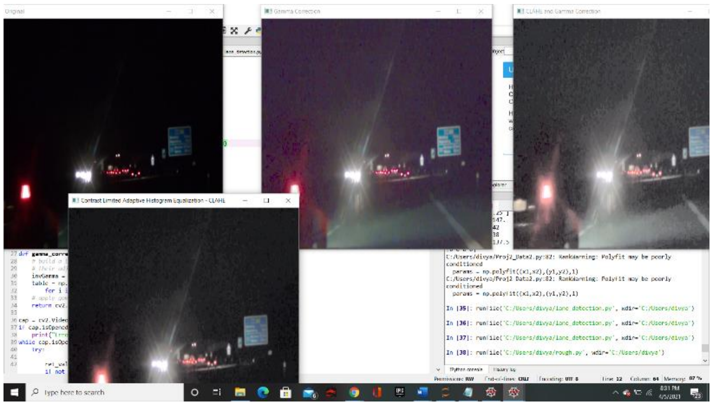
April 5th, 2021

Table of Contents

A1) Histogram Equalization :

- In the given video, the contrast was extremely dark that means pixels of certain intensity have very high frequency as opposed to other pixels.
- Normal histogram Equalization cumulative frequency distribution was applied which tries to equalize the values by equalizing the distribution of the peaked intensities as CFD always increases and the intensity are divided by the number of its pixels.
- This did not give a very good result and to mitigate this problem gamma correction was applied.
- The monitor provides linear combination for intensities to generate different color to the human eye. In actual the voltage input has a non-linear relation which is bound by a constant gamma factor. This in turn was rectified by dividing each pixel with a constant gamma value to make it linear again. On this equalization was applied. The self-written function was again not giving good values hence, inbuilt function was used for histogram equalization. This was compared with clahe correction as well.

Future work : Try to apply gamma correction in HLS space.

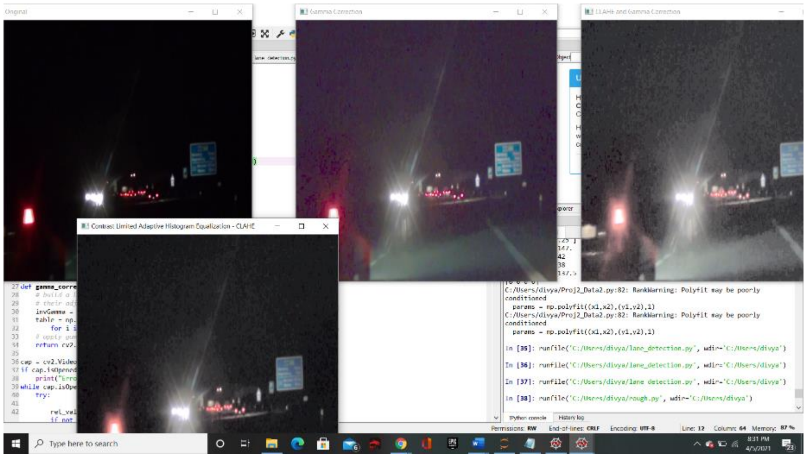

Original
Gamma Correction
CLAHE and Gamma Correction
Contrast Limited Adaptive Histogram Equalization - CLAHE
IPython console
History log
Type here to search

```
def gamma_corre
    # build a l
    # their adj
    invGamma -
    table - np.
        for i i
    # apply gam
    return cv2.

cap - cv2.Video
if cap.isOpened
    print("Erro
while cap.isOpe
    try:

        ret_val
        if not
```

```
.25 ]
147.
42
38
137.5
C:/Users/divya/Proj2_Data2.py:82: RankWarning: Polyfit may be poorly
conditioned
  params = np.polyfit((x1,x2),(y1,y2),1)
C:/Users/divya/Proj2_Data2.py:82: RankWarning: Polyfit may be poorly
conditioned
  params = np.polyfit((x1,x2),(y1,y2),1)

In [35]: runfile('C:/Users/divya/lane_detection.py', wdir='C:/Users/divya')

In [36]: runfile('C:/Users/divya/lane_detection.py', wdir='C:/Users/divya')

In [37]: runfile('C:/Users/divya/lane_detection.py', wdir='C:/Users/divya')

In [38]: runfile('C:/Users/divya/rough.py', wdir='C:/Users/divya')
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 12 Column: 64 Memory: 87 %

8:31 PM 4/5/2021


Original
Gamma Correction
CLAHE and Gamma Correction
Contrast Limited Adaptive Histogram Equalization - CLAHE
IPython console
History log
Type here to search

```
def gamma_corre
    # build a l
    # their adj
    invGamma -
    table - np.
        for i i
    # apply gam
    return cv2.

cap - cv2.Video
if cap.isOpened
    print("Erro
while cap.isOpe
    try:

        ret_val
        if not
```

```
.25 ]
147.
42
38
137.5
C:/Users/divya/Proj2_Data2.py:82: RankWarning: Polyfit may be poorly
conditioned
  params = np.polyfit((x1,x2),(y1,y2),1)
C:/Users/divya/Proj2_Data2.py:82: RankWarning: Polyfit may be poorly
conditioned
  params = np.polyfit((x1,x2),(y1,y2),1)

In [35]: runfile('C:/Users/divya/lane_detection.py', wdir='C:/Users/divya')

In [36]: runfile('C:/Users/divya/lane_detection.py', wdir='C:/Users/divya')

In [37]: runfile('C:/Users/divya/lane detection.py', wdir='C:/Users/divya')

In [38]: runfile('C:/Users/divya/rough.py', wdir='C:/Users/divya')
```

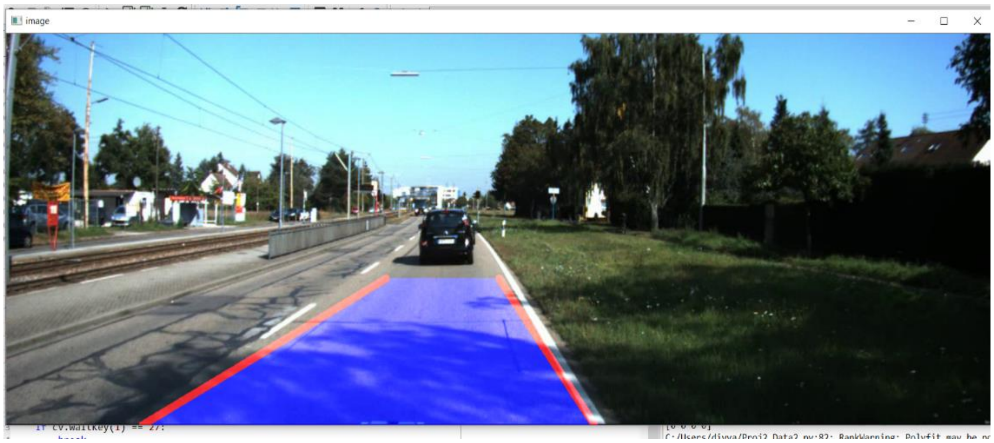
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 12 Column: 64 Memory: 87 %

8:31 PM 4/5/2021

A2) LANE DETECTION

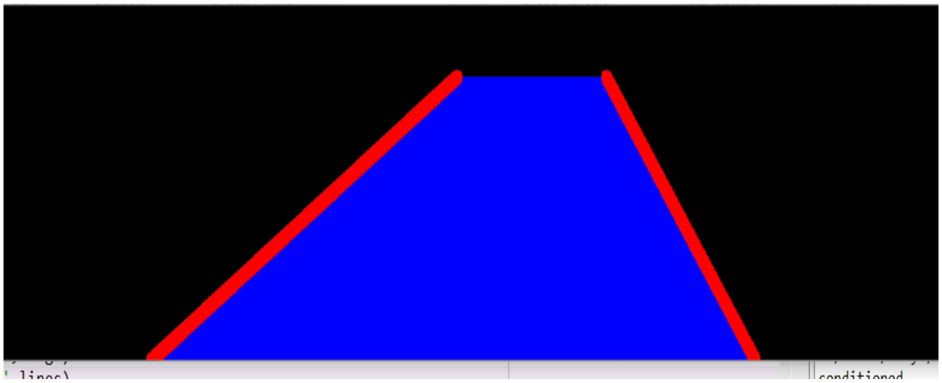
Steps:

- Undistorted image using camera intrinsic matrix and distortion coefficient.
- Smoothened the image using Gaussian filter. (Gaussian filter, mean filter and bilinear filter were tried and gaussian filter with a kernel of (7,7) gave the best results)
- Image was cropped for region of interest according to the allowed limit which was cropping the sky.
- The image was turned into gray scale to get just one channel for applying threshold.
- Binary threshold was applied to with a range of (230,255) to get the white lanes specifically close to the camera as other white images were also majorly ignored.
- Canny edge detection was applied which internally uses optimized sobel operator to get single lines along gradient changes. Still multiple edge lines were acquired as shown in figure. This was resolved in further steps.
- All the lines were segregated to left and right lines according to their slopes. As it's an image in 2D project the parallel lines are shown as a triangle hence left line will always have its slope > 0 and right line will always have its slope < 0. The slope and intercept were averaged, and a new line was drawn using that information. ($y = mx + c$)
- The lanes were displayed on the cropped image using bitwise function and was overlayed onto the original image.



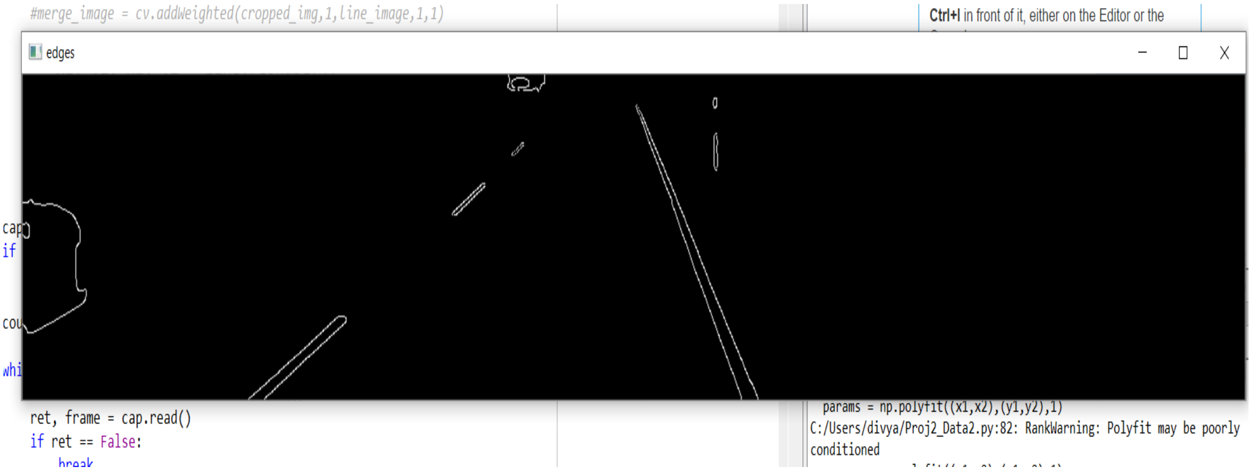
```
if cv.waitKey(1) == 27:
```

```
C:/Users/divya/Proj2_Data2.py:82: RankWarning: Polyfit may be po
```



```
lines)
```

```
conditioned
```

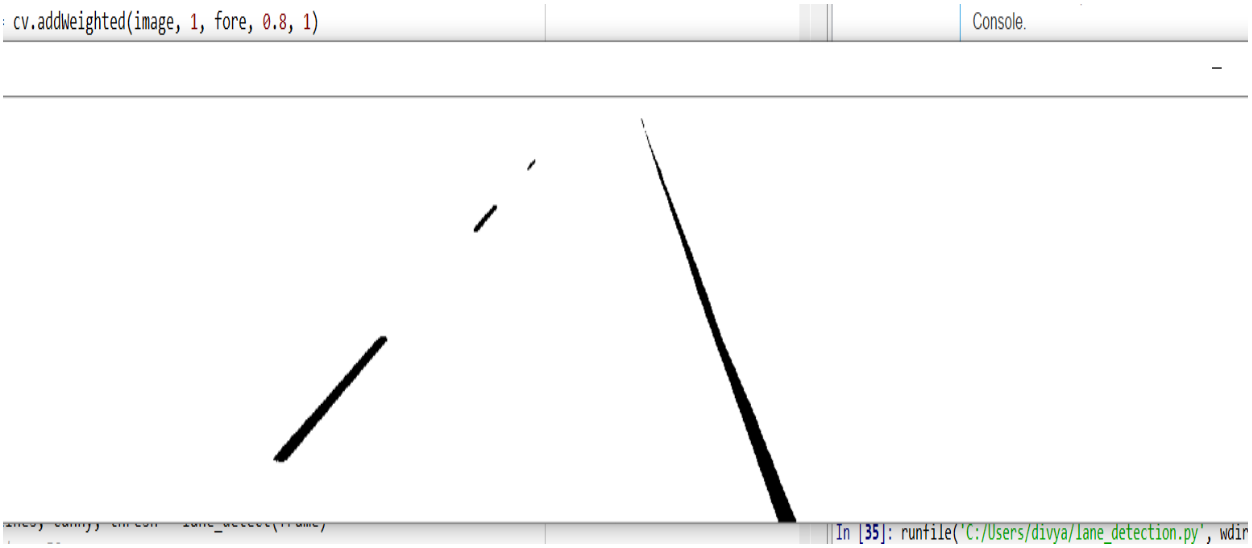


```
#merge_image = cv.addWeighted(cropped_img,1,line_image,1,1)
```

Ctrl+I in front of it, either on the Editor or the

```
ret, frame = cap.read()
if ret == False:
    break
```

```
params = np.polyfit((x1,x2),(y1,y2),1)
C:/Users/divya/Proj2_Data2.py:82: RankWarning: Polyfit may be poorly
conditioned
```



```
cv.addWeighted(image, 1, fore, 0.8, 1)
```

```
lines, canny, thresh = lane_detect(frame)
```

```
In [35]: runfile('C:/Users/divya/lane_detection.py', wdir
```

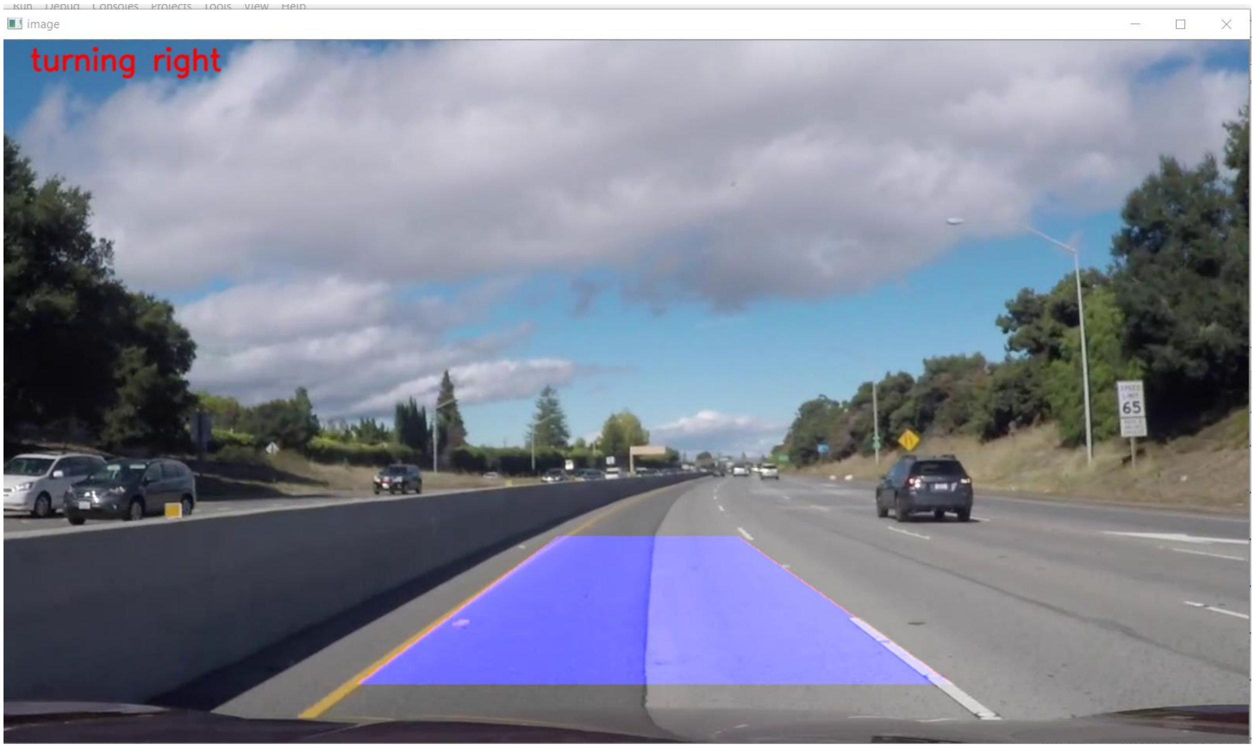
A2) LANES DETECTION and TURN PREDICTION on Challenge Video.

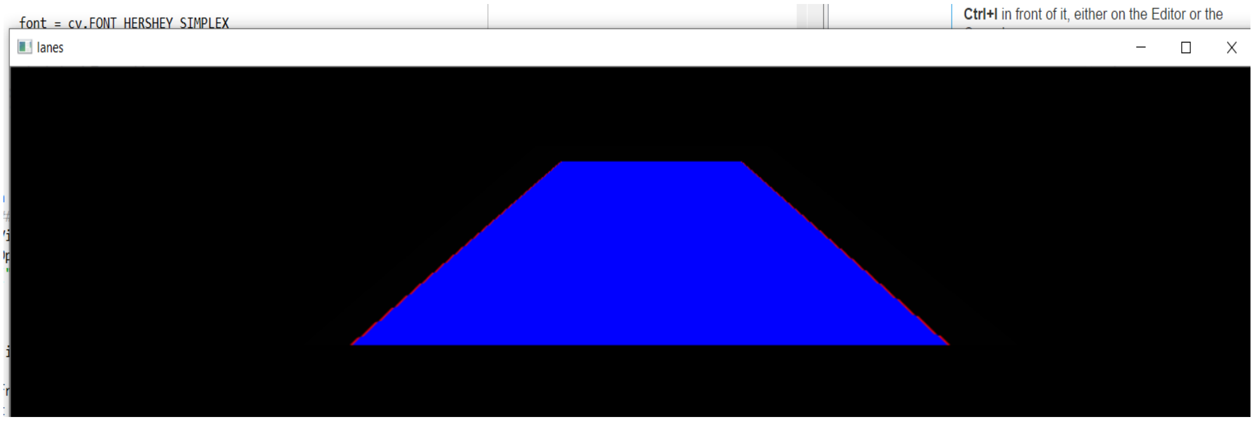
- Similar approach was tried for challenge video but it did not give very good result as there were too much disturbance from other white sources with an added noise of the diamond structure in the middle of the lane
- Thus, after following the similar approach until smoothening the image four points were chosen as a rectangle from the lane onto which warping was applied using homography. This had two benefits. First, it gave bird eye view which made it easier to process, either fitting the curve or applying homography, second it gave a particular region of interest which was highly in this video.
- A combined mask was applied as shown in the images below for yellow and white and were added using bitwise and operation.
- In a similar way of problem 1 the hough lines were applied but in this case they could not be differentiated into left or right lines using slope. Hence, in this scenario intercept was used to segregate which further allowed to ignore the middle part of the lane and purely gave the detected lane.
- To predict the slope the difference was taken between left lane and right lane. The difference was between right and left as right fluctuates to a positive curve when the right turn is taken. When the difference is close to zero or negative the car is predicted to take a right turn and if it a high

positive value (left – (-right)) , it is predicted to take a left turn. In case of equal slope, it is predicted to go straight.

- This was further put into text using opencv inbuilt function to demonstrate it on the image.
- The overlay process was similar to problem 1.

Future Work: Includes applying moving average to make up for the lost images.





```
font = cv.FONT_HERSHEY_SIMPLEX
```

Ctrl+I in front of it, either on the Editor or the

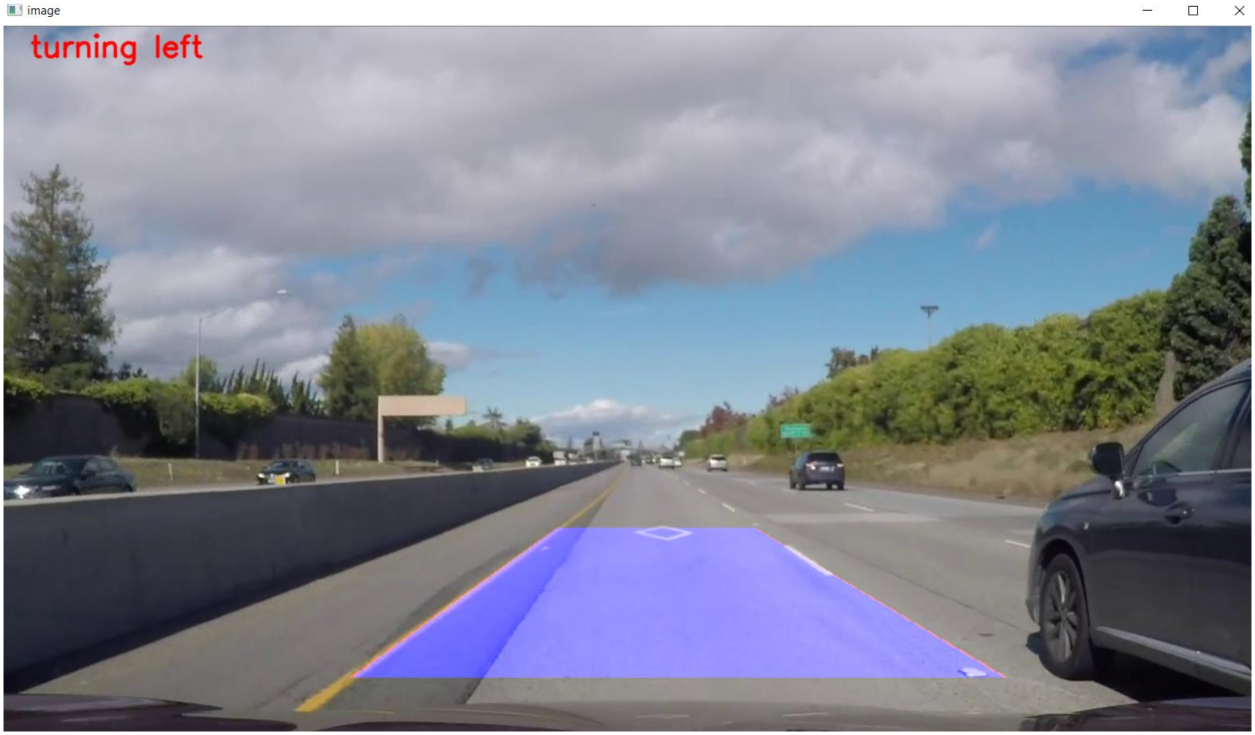
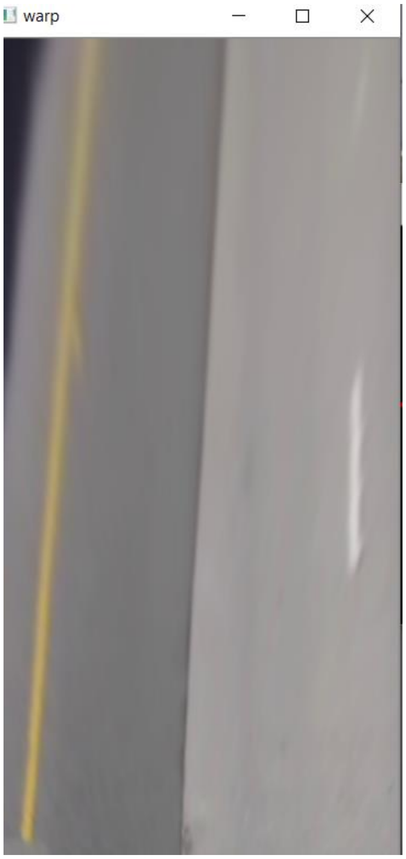
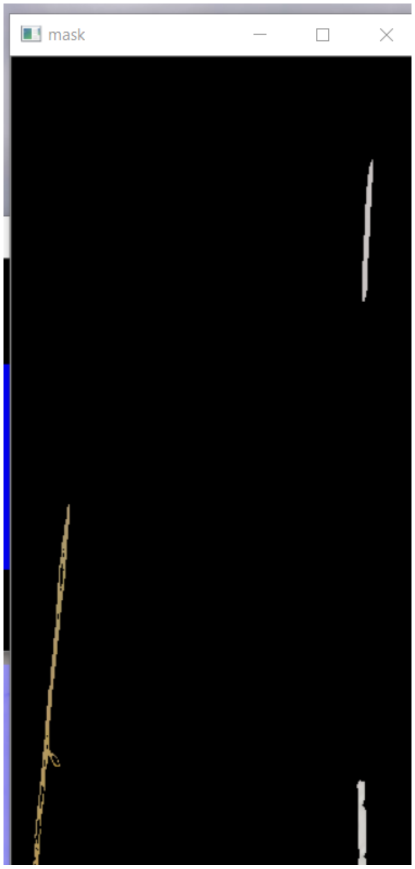


image
turning left


warp


mask

##