**Finding Lane Lines on the Road**

**Writeup Template**

**You can use this file as a template for your writeup if you want to submit it as a markdown file. But feel free to use some other method and submit a pdf if you prefer.**

**Finding Lane Lines on the Road**

The goals / steps of this project are the following:

* Make a pipeline that finds lane lines on the road
* Reflect on your work in a written report

**Reflection**

**1. Describe your pipeline. As part of the description, explain how you modified the draw\_lines() function.**

Draw\_line()함수를 적용하기 전까지 이미지 프로세싱 파이프라인은 다음과 같은 순서로 구성 하였다.

(RGB2Gray-> equalizeHist-> GaussianBlur-> Canny-> HoughLineP)

RGB2Gray: 칼라 이미지를 흑백 이미지로 변환하는 함수 이다.

equalizeHist: 이미지의 콘트라스트를 향상 시킨다.

GaussianBlur: 가우시안 노이즈를 제거하는데 효과적이다.

Canny: 이미지에서 경계면을 추출하기 위한 함수

HoughLineP: Canny Edge Detection 이후에 같은 기울기를 가진 특징을 추출하기 위함(도로 선은 같은 기울기를 가질 것이기 때문이다.)

이미지 프로세싱을 거치고 난 후 도로선 추출 결과는 다음과 같다.

(그림1)

하지만, 본 그림을 아래 그림과 같이 선을 이어서 표현하기 위해서는 draw\_lines( ) 함수를 수정할 필요가 있다.

(그림2)모범 답안(‘laneLines\_thirdPass’)

draw\_lines( ) 함수는 다음과 같이 수정하였다.

draw\_lines( ) 함수: cv2.HoughLinesP() 함수를 통해 나온 x1, y1, x2, y2 좌표들을 통해( (그림1)에서 추출한 도로상 선들의 x,y 좌표 값) 그림2 와 같이 선을 표현하는 것이 목적인 함수이다. 즉, 차선 검출을 더 견고하게 표현하는 것이 목적.

먼저 왼쪽 차선과 오른쪽 차선을 분리해서 리스트 자료형에 저장하였고, 왼쪽 차선의 (Left\_bottem\_x, Left\_x + Left\_low\_y, Left\_high\_y)좌표를 추출하였다.

마찬 가지로 오른쪽 차선의 (Right\_bottem\_x, Right \_x + Right \_low\_y, Right \_high\_y) 좌표를 추출하였다.

이후, cv2.line(img, (Left\_bottem\_x, Left\_low\_y), (Left\_x, Left\_high\_y), color, thickness)

cv2.line(img, (Right\_bottem\_x, Right \_low\_y), (Right \_x, Right \_high\_y), color, thickness)

함수를 통해 선을 그려 주었다.

(그림3)draw\_lines()함수 적용 후

다음은 이미지 프로세싱과 draw\_lines( ) 함수를 거치면서 이미지가 어떻게 변하는지 보여준다.

※ 이미지 전처리 변화

(RGB2Gray-> equalizeHist-> GaussianBlur-> Canny-> HoughLineP-> draw\_lines( )함수)

원본 이미지(‘whiteCarLaneSwitch.jpg’)



RGB2Gray



equalizeHist



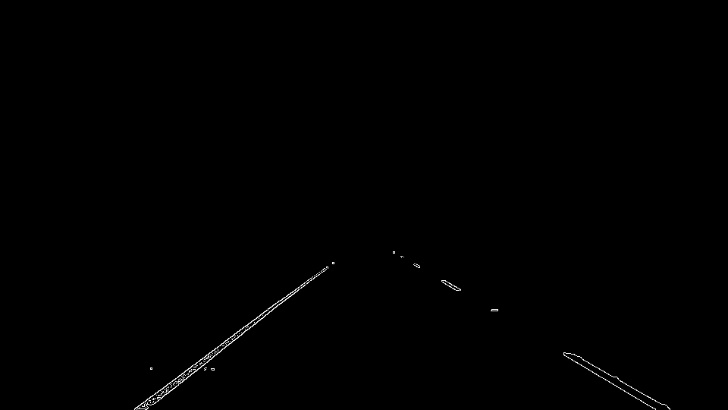
GaussianBlur: 큰 차이가 없다.



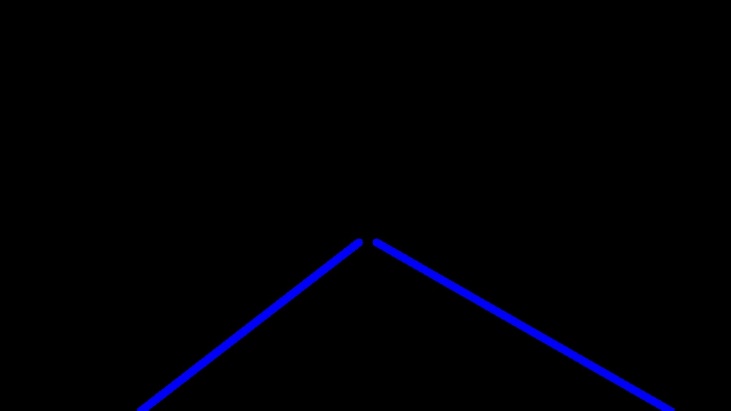
Canny Edge Detection



HoughLineP



draw\_lines( ) 함수 적용 후



**2. Identify potential shortcomings with your current pipeline**

제안하는 파이프라인의 잠재적 단점은 다음과 같은 것을 들 수 있다.

1. 쉐도우(언덕에 의한 그림자, 구름에 의한 그림자 등…)가 이미지에 나타날 때, 라인을 제대로 추출할 수 있는 지는 확인해 보지 못했다.

해결책: 그림자가 적용된 이미지 데이터를 수집하여 라인 추출이 제대로 일어나는 지 확인할 필요가 있다.

2.

Another shortcoming could be ...

**3. Suggest possible improvements to your pipeline**

A possible improvement would be to ...

Another potential improvement could be to ...