✔️

[Self-Driving] P4. Advanced lane finding

링크	https://github.com/ndrplz/self-driving-car/tree/master/project_4_advanced_lane_finding
Category	Self-Driving Study
Keywords	Calibration Sobel detection lane finding
상위 개념	
하위 개념	✔️ Morphological Closing (Opening), ✔️ Sobel Kernel
속성	
Subcategory	개념 정리

Object

- 왜곡 이미지를 원복하기
- color transformation, gradient를 사용해서 thresholded binary image 생성
- Apply a perspective transform to rectify binary image ("birds-eye view").
- 휘어진 lane를 자동차 위치에 맞춰서 캘

Camera Calibration

OpenCV에서 여러가지 method를 제공해준다

- `cv2.findChessboardCorners(img, pattern_size)`

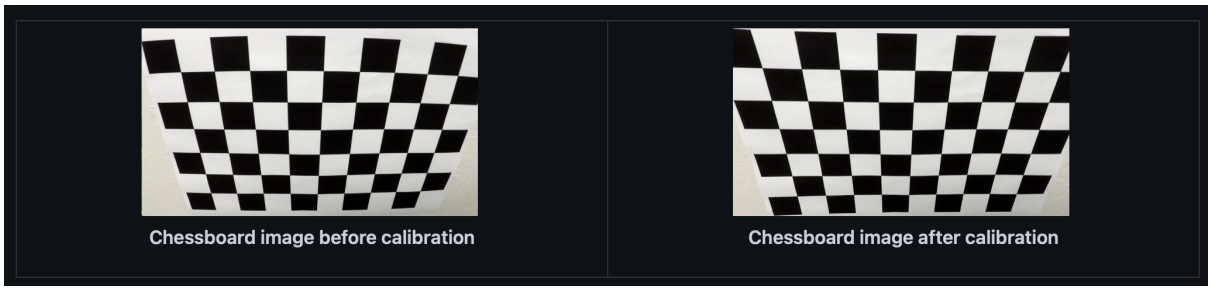
  Corner를 발견하면 반환 모서리 array 반환

- `cv2.calibrationCamera()`

카메라 행렬, 왜곡 정보를 반환

- `cv2.undistort()`

  왜곡 보정



Pipeline

1. cal 과정을 통해 얻은 Camera matrix와 Distortion coefficient를 통해 이미지를 보정한 후에

   도로 이미지를 공부해야한다.

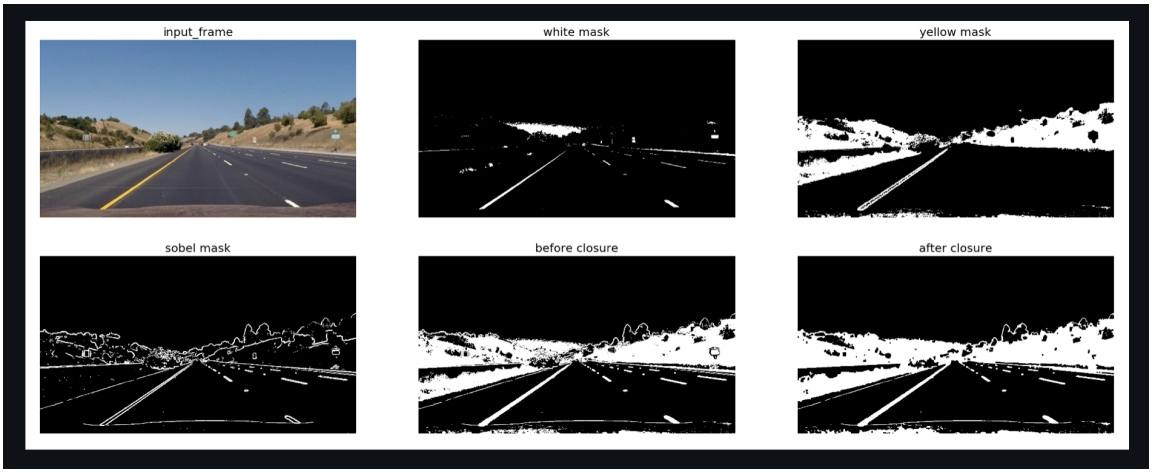
2. **제일 먼저 Input에 대해 Binary Image로 변환하는 것이 Lane detection의 첫 시작이다.**

   먼저 Image의 Histogram에 대해 **Histogram Equalization**을 진행한다.

   그 후 Yellow Lane를 검출하기 위해 HSV채널에서 V(명도)를 이용하고

   Sobel Kernel을 이용해서 Line Detection을 한다.

   이렇게 검출한 Line 사이의 Gap을 채우기 위하여 Morphological Closing을 진행한다.



3. **Perspective Tranformation을 진행한다.**

Perspective Tranformation(:= Homography)은 4쌍의 각각 대응하는 점만 있으면 구할 수 있다.

위에서는 중심 두개와 하단 두개를 잡았다.

이 과정에서 변환 행렬이 필요한데 이는 `cv2.getPerspectiveTransform` 을 통해 구할 수 있다.

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{00} & b_0 \\ a_{10} & a_{11} & b_1 \\ a_{20} & a_{21} & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

4. **이제 Binary Image에서 어느 부분이 Lane인지 판별을 해야한다.**

이미지의 하단에서 두 개의 Window를 만들고 상단으로 이동하면서 Lane을 추적한다.

이는 Histogram의 값이 높은 지점를 탐색하면서 진행한다.

만약 Image가 아니라 Video의 경우라면 이전 Frame에서의 Lane 위치를 가져오면서 탐색한다. 이는 fps가 어느정도 높다면 연속된 frame 사이에서 lane의 위치는 크게 차이가 없을 것이기 때문.

5. **Lane의 곡률을 구하고 중심을 기준으로 vehicle의 위치를 판단한다.**

   Lane의 center는 좌lane 우lane의 평균을 통해 구한다.

   `np.polyfit(x,y,dim)` 을 통해 3개의 coefficient를 알 수 있는데(2차계수 1차계수 bias),

   이를 통해 곡률을 구한다.

6. 위의 과정을 de-wrapping하여 원본 이미지에 반영