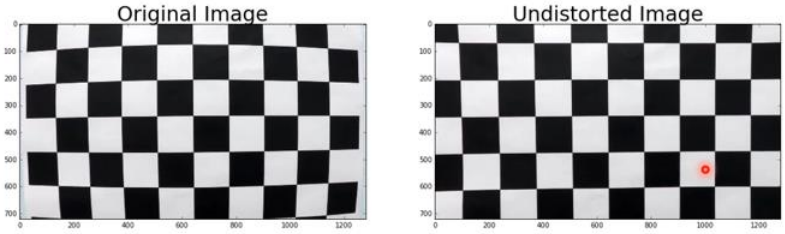
- 원근 변환과 슬라이딩 윈도우를 이용한 차선 찾기

1. Camera Calibration
2. Bird's eye view
3. 이미지 임계 값 및 이진 이미지
4. 슬라이딩 윈도우로 차선 위치 파악
5. 파악된 차선 위치 원본 이미지에 표시

➔ Camera Calibration - 카메라 보정

- ■ 카메라가 곡면 렌즈를 사용하여 이미지를 형성하고 이로 인해 가장 자리가 왜곡되어 보이는 현상을 해결하기 위한 방법
- ■ 왜곡된 지점을 왜곡되지 않는 지점으로 Mapping하여 왜곡을 없앰

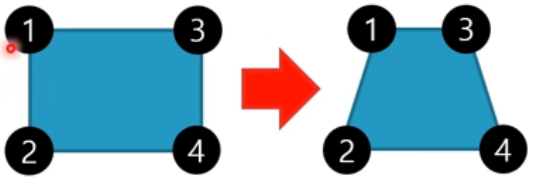
• 체스 판으로 Camera Calibration 진행 시

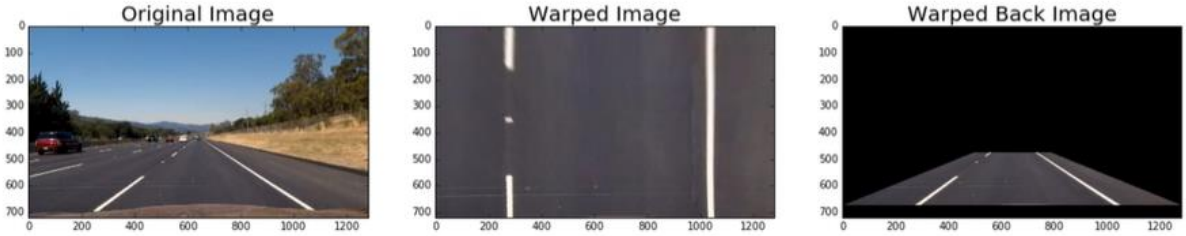


➔ Bird's eye View

- ■ 새가 하늘에서 내려다보는 듯한 구도로 위에서 아래를 내려다보는 방식
- ■ 선의 곡률을 측정하기 위해 도로 이미지를 하향식 보기로 변환

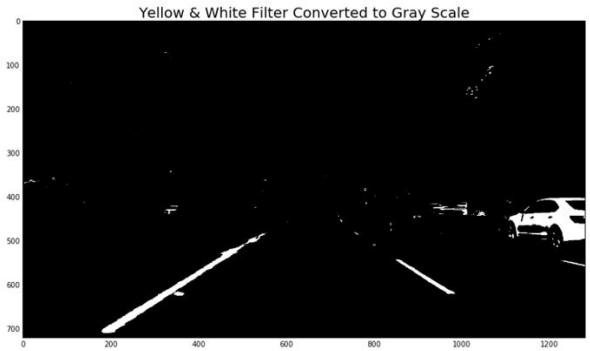
```
# 변환 전과 후의 기준이 되는 4개 점의 좌표값을 지정
# 점은 좌상 → 좌하 → 우상 → 우하

pts1 = np.float32( 이동전 4점의 좌표 )

pts2 = np.float32( 변환후 4점의 좌표)

M = cv2.getPerspectiveTransform(pts1, pts2)

dst = cv2.warpPerspective(img, M, (cols,rows))
```





➔ 이미지 임계 값 및 이진 이미지

- 차선이 명확하게 보이는 이미지를 생성하기 위해 색상 임계 값 조절
- 이미지를 흰색과 노란색으로 마스킹
- Gray scale로 변환
- 이진 이미지 생성



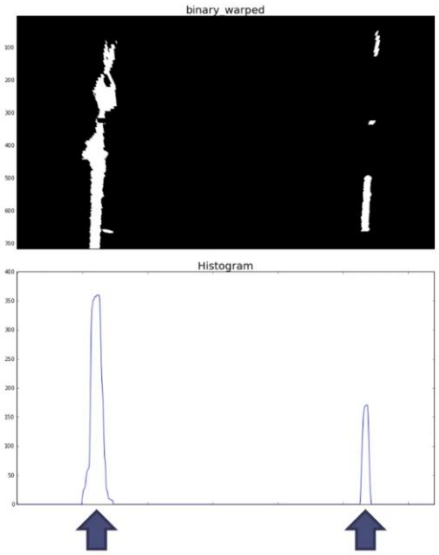
- HSV(Hue Saturation Value) : 명도가 낮을수록 검은색, 명도가 낮고 채도가 낮을수록 흰색
- LAB(Lightness Red/green Yellow/Blue) : 노란색 차선을 인식할 때 B를 사용하면 좋은 성능을 냄
- HLS(Hue Lightness, Saturation) : 흰색 차선을 인식할 때 L을 사용하면 좋은 성능을 냄

➔ 차선 식별

■ 히스토그램 방법

도로 이미지에 보정, 임계 값 및 원근 변환을 적용하여 차선이 두드러지는 이진 이미지를 얻음

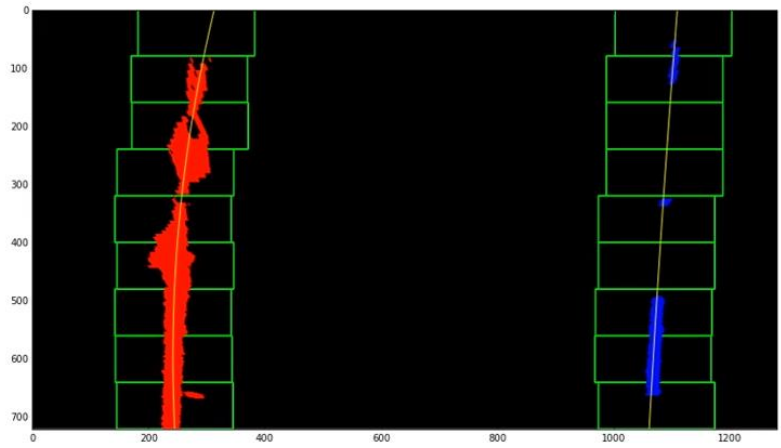
- ◆ 얻은 이미지에서 어떤 픽셀이 라인의 일부이고, 왼쪽 또는 오른쪽 라인인지 결정
- ◆ 각 열에 따라 픽셀 개수를 더하면 히스토그램에서 가장 눈에 띄는 두개의 peak가 생성되고, 차선의 x 위치를 파악할 수 있음



■ 슬라이딩 윈도우

- ◆ 선 중심 주변에 배치된 슬라이딩 윈도우를 사용하여 프레임 상단까지 선을 찾아 따라 감

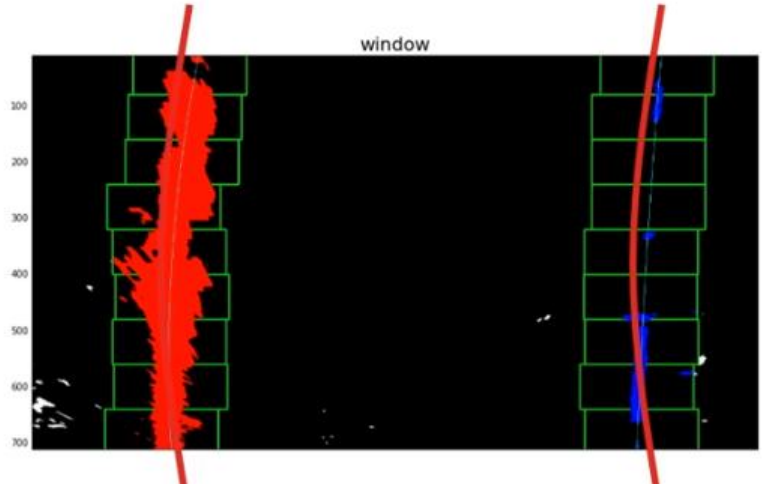
◆ 한 윈도우 안에서 감지되는 선의 중심을 기준으로 계속 윈도우가 쌓임



◆ 슬라이딩 윈도우가 여러 개 쌓이게 되면, 그 중심을 연결하여 선을 그림

◆ Polyfit을 사용하여 2차원으로 표현 -> ay^2+by+c=x

◆ 2차식을 통해 a, b, c 값을 구함



■ **차선 영역을 녹색으로 칠하기**

➔ 파악된 차선 위치 원본 이미지에 표시