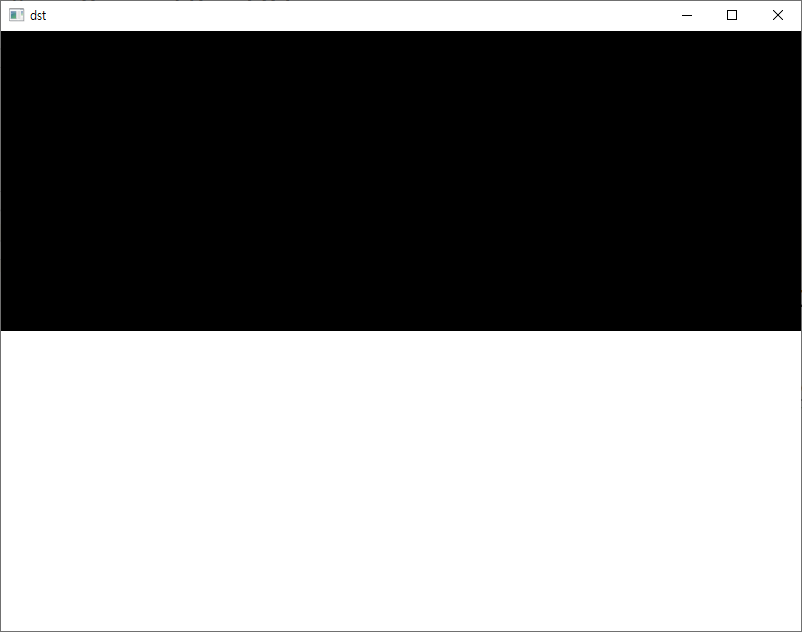
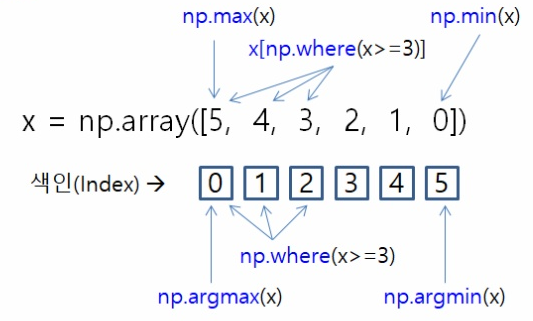
| import cv2  import numpy as np  img\_zeros = np.zeros((600,800,3), dtype=np.uint8)  height, width = img\_zeros.shape[:2]  img\_zeros[height//2:,:]=255 # 이미지의 height의 절반부터 255로 만듬  cv2.imshow('dst',img\_zeros)  cv2.waitKey(0)  cv2.destroyAllWindows() |
| --- |



ArgMax함수 : 최대값의 좌표 리턴

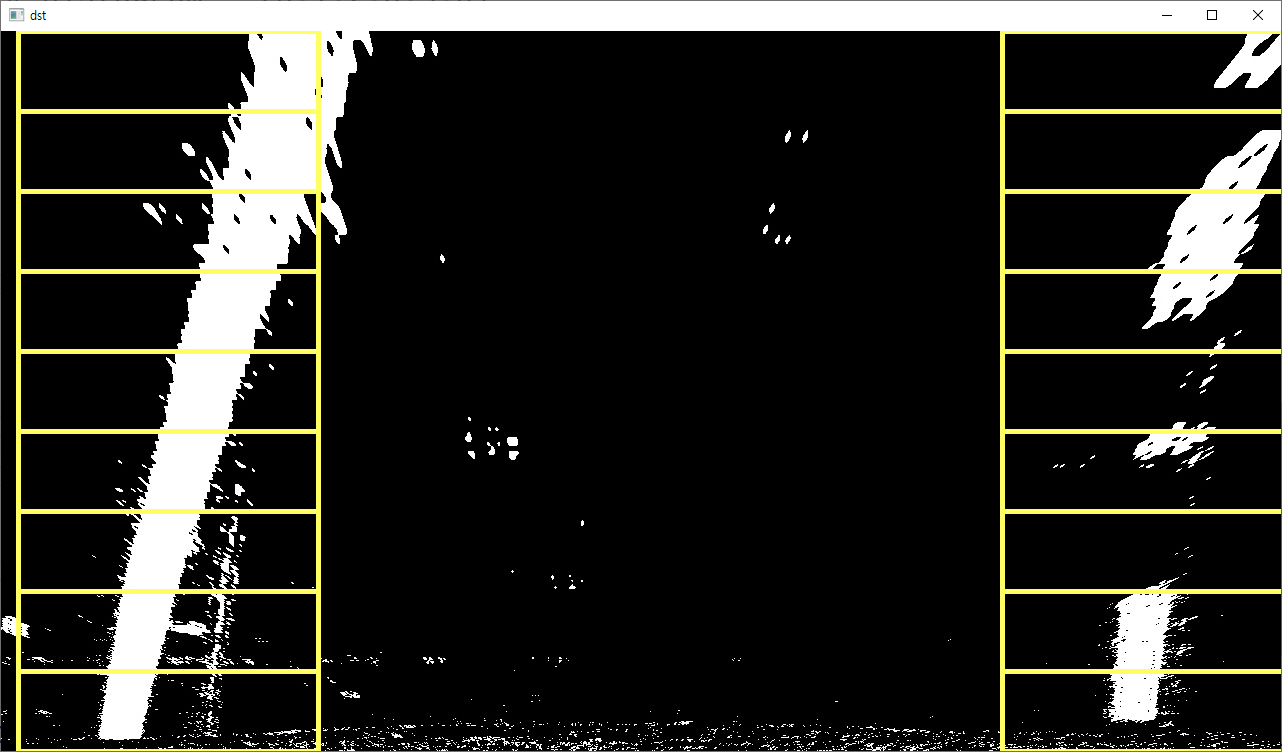


| import numpy as np  histogram = [0,1,2,3,4,5]  midpoint = len(histogram)//2  # midpoint = histogram.shape[0] // 2  # 1 2 3  leftx\_base = np.argmax(histogram[:midpoint])  # 4 5 6  rightx\_base = np.argmax(histogram[midpoint:])+midpoint  print(leftx\_base,rightx\_base) |
| --- |

| 2 5 |
| --- |

윈도우 그리기 # 1

| import cv2  import numpy as np  nwindows = 9  margin = 150  img\_src = cv2.imread('images/warped\_image.jpg',cv2.IMREAD\_COLOR)  img\_binary = cv2.cvtColor(img\_src,cv2.COLOR\_BGR2GRAY)  height, width = img\_binary.shape[:2]  # 이미지의 아래쪽 절반의 column의 합(|||||)을 리스트로  histogram = np.sum(img\_binary[height//2:,:], axis=0)  midpoint = len(histogram)//2  # histogram에서 좌우측의 최대값의 좌표를 구함  leftx\_base = np.argmax(histogram[:midpoint])  rightx\_base = np.argmax(histogram[midpoint:])+midpoint  # 여러개의 윈도우로 자를때 각 윈도우의 높이  window\_height = int(height/nwindows)  leftx\_current = leftx\_base  rightx\_current = rightx\_base  for window in range(nwindows):  win\_y\_low = height - (window+1)\*window\_height  win\_y\_high = height - window\*window\_height  win\_xleft\_low = leftx\_current - margin  win\_xleft\_high = leftx\_current + margin  win\_xright\_low = rightx\_current - margin  win\_xright\_high = rightx\_current + margin  cv2.rectangle(img\_src, (win\_xleft\_low,win\_y\_low),  (win\_xleft\_high,win\_y\_high),(100,255,255), 3)  cv2.rectangle(img\_src, (win\_xright\_low, win\_y\_low),  (win\_xright\_high, win\_y\_high),(100,255,255), 3)  cv2.imshow('dst',img\_src)  cv2.waitKey(0)  cv2.destroyAllWindows() |
| --- |



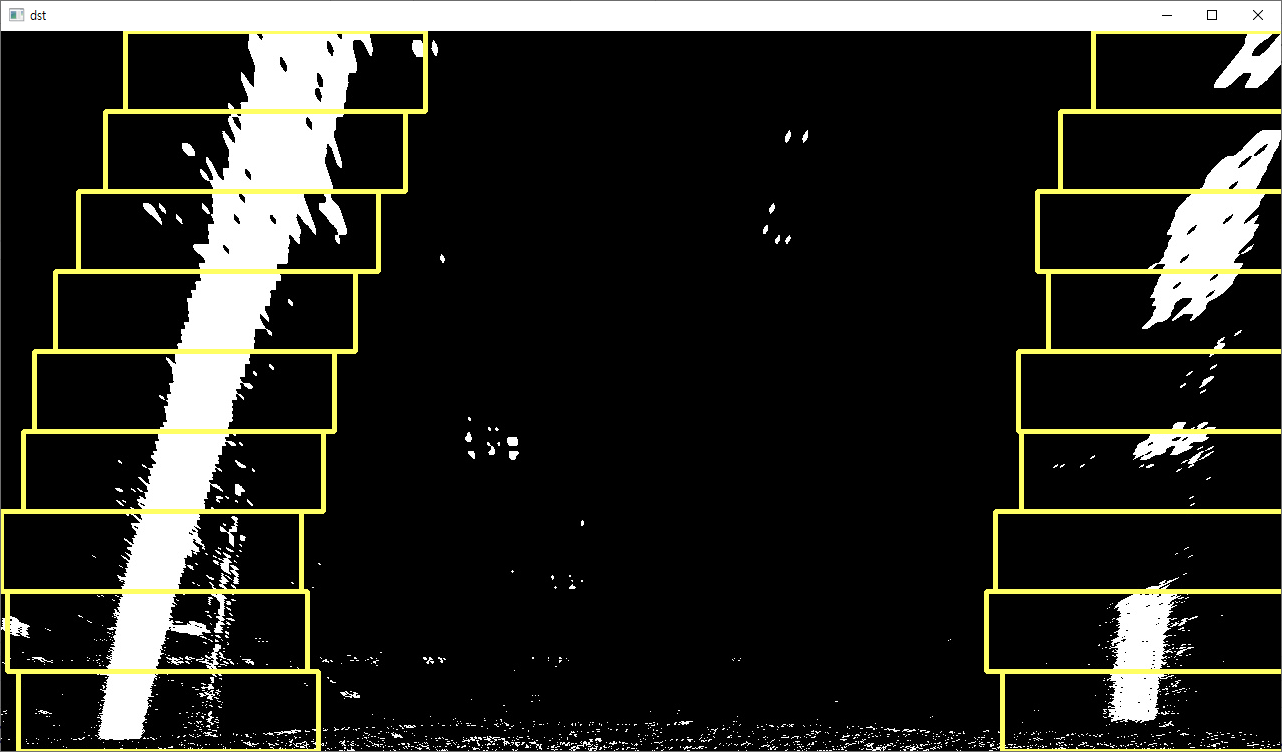
NON-Zero

| import numpy as np  # 이미지 가운데 0이아닌 값의 좌표값을 리턴함  img = np.array([[0,1,2],[3,4,0]])  nonzero = img.nonzero()  nonzero\_y = np.array(nonzero[0])  nonzero\_x = np.array(nonzero[1])  print(nonzero\_y)  print(nonzero\_x) |
| --- |

| [0 0 1 1]  [1 2 0 1] |  |
| --- | --- |

사각형 윈도우 업데이트

| import cv2  import numpy as np  nwindows = 9  margin = 150  minpix = 1  img\_src = cv2.imread('images/warped\_image.jpg',cv2.IMREAD\_COLOR)  img\_binary = cv2.cvtColor(img\_src,cv2.COLOR\_BGR2GRAY)  height, width = img\_binary.shape[:2]  histogram = np.sum(img\_binary[height//2:,:], axis=0)  midpoint = len(histogram)//2  leftx\_base = np.argmax(histogram[:midpoint])  rightx\_base = np.argmax(histogram[midpoint:])+midpoint  window\_height = int(height/nwindows)  # 영상에서 0이 아닌 모든 점의 x,y점을 정의  nonzero = img\_binary.nonzero()  nonzero\_y = np.array(nonzero[0])  nonzero\_x = np.array(nonzero[1])  leftx\_current = leftx\_base  rightx\_current = rightx\_base  #왼쪽, 오른쪽 차선의 조건을 만족하는 nonzero index를 받기 위해 리스트 생성  left\_lane\_inds = []  right\_lane\_inds = []  for window in range(nwindows):  win\_y\_low = height - (window+1)\*window\_height  win\_y\_high = height - window\*window\_height  win\_xleft\_low = leftx\_current - margin  win\_xleft\_high = leftx\_current + margin  win\_xright\_low = rightx\_current - margin  win\_xright\_high = rightx\_current + margin  cv2.rectangle(img\_src, (win\_xleft\_low,win\_y\_low),  (win\_xleft\_high,win\_y\_high),(100,255,255), 3)  cv2.rectangle(img\_src, (win\_xright\_low, win\_y\_low),  (win\_xright\_high, win\_y\_high),(100,255,255), 3)  # 아래 조건을 만족하는 점들의 인덱스 값을 리턴함  good\_left\_inds = ((nonzero\_y >= win\_y\_low) &  (nonzero\_y < win\_y\_high) &  (nonzero\_x >= win\_xleft\_low) &  (nonzero\_x < win\_xleft\_high)).nonzero()[0]  good\_right\_inds = ((nonzero\_y >= win\_y\_low) &  (nonzero\_y < win\_y\_high) &  (nonzero\_x >= win\_xright\_low) &  (nonzero\_x < win\_xright\_high)).nonzero()[0]  # 리스트에 조건을 만족하는 인덱스 값을 append  left\_lane\_inds.append(good\_left\_inds)  right\_lane\_inds.append(good\_right\_inds)  # If you found > minpix 픽셀의 개수가 minpix보다 크면 사각형의 센터값 업데이트  if len(good\_left\_inds) > minpix:  leftx\_current = int(np.mean(nonzero\_x[good\_left\_inds]))  if len(good\_right\_inds) > minpix:  rightx\_current = int(np.mean(nonzero\_x[good\_right\_inds]))  cv2.imshow('dst',img\_src)  cv2.waitKey(0)  cv2.destroyAllWindows() |
| --- |



1차원 배열 합치기

| import numpy as np  arrays = [np.array([1,2,3]),np.array([5,6,7])]  # arrays = [[1,2,3], [5,6,7]]  print(arrays)  arrays = np.concatenate(arrays)  print(arrays) |
| --- |

| [array([1, 2, 3]), array([5, 6, 7])]  [1 2 3 5 6 7] |
| --- |

NonZero 추출하여 선그리기

| import cv2  import numpy as np  nwindows = 9  margin = 150  minpix = 1  left\_a, left\_b, left\_c = [],[],[]  right\_a, right\_b, right\_c = [],[],[]  img\_src = cv2.imread('images/warped\_image.jpg',cv2.IMREAD\_COLOR)  img\_binary = cv2.cvtColor(img\_src,cv2.COLOR\_BGR2GRAY)  height, width = img\_binary.shape[:2]  left\_fit\_= np.empty(3)  right\_fit\_ = np.empty(3)  histogram = np.sum(img\_binary[height//2:,:], axis=0)  midpoint = len(histogram)//2  leftx\_base = np.argmax(histogram[:midpoint])  rightx\_base = np.argmax(histogram[midpoint:])+midpoint  window\_height = int(height/nwindows)  # 영상에서 0이 아닌 모든 점의 x,y점을 정의  nonzero = img\_binary.nonzero()  nonzero\_y = np.array(nonzero[0])  nonzero\_x = np.array(nonzero[1])  leftx\_current = leftx\_base  rightx\_current = rightx\_base  #왼쪽, 오른쪽 차선의 조건을 만족하는 nonzero index를 받기위해 리스트 생성  left\_lane\_inds = []  right\_lane\_inds = []  for window in range(nwindows):  win\_y\_low = height - (window+1)\*window\_height  win\_y\_high = height - window\*window\_height  win\_xleft\_low = leftx\_current - margin  win\_xleft\_high = leftx\_current + margin  win\_xright\_low = rightx\_current - margin  win\_xright\_high = rightx\_current + margin  cv2.rectangle(img\_src, (win\_xleft\_low,win\_y\_low),  (win\_xleft\_high,win\_y\_high),(100,255,255), 3)  cv2.rectangle(img\_src, (win\_xright\_low, win\_y\_low),  (win\_xright\_high, win\_y\_high),(100,255,255), 3)  # 아래 조건을 만족하는 점들의 인덱스 값을 리턴함  good\_left\_inds = ((nonzero\_y >= win\_y\_low) &  (nonzero\_y < win\_y\_high) &  (nonzero\_x >= win\_xleft\_low) &  (nonzero\_x < win\_xleft\_high)).nonzero()[0]  good\_right\_inds = ((nonzero\_y >= win\_y\_low) &  (nonzero\_y < win\_y\_high) &  (nonzero\_x >= win\_xright\_low) &  (nonzero\_x < win\_xright\_high)).nonzero()[0]  # 리스트에 조건을 만족하는 인덱스 값을 append  left\_lane\_inds.append(good\_left\_inds)  right\_lane\_inds.append(good\_right\_inds)  # If you found > minpix 픽셀의 개수가 minpix보다 크면 사각형의 센터값 업데이트  if len(good\_left\_inds) > minpix:  leftx\_current = int(np.mean(nonzero\_x[good\_left\_inds]))  if len(good\_right\_inds) > minpix:  rightx\_current = int(np.mean(nonzero\_x[good\_right\_inds]))  # Concatenate the arrays of indices  left\_lane\_inds = np.concatenate(left\_lane\_inds)  right\_lane\_inds = np.concatenate(right\_lane\_inds)  # Extract left and right line pixel positions  leftx = nonzero\_x[left\_lane\_inds]  lefty = nonzero\_y[left\_lane\_inds]  rightx = nonzero\_x[right\_lane\_inds]  righty = nonzero\_y[right\_lane\_inds]  # Fit a second order polynomial to each  left\_fit = np.polyfit(lefty, leftx, 2)  right\_fit = np.polyfit(righty, rightx, 2)  left\_a.append(left\_fit[0])  left\_b.append(left\_fit[1])  left\_c.append(left\_fit[2])  right\_a.append(right\_fit[0])  right\_b.append(right\_fit[1])  right\_c.append(right\_fit[2])  left\_fit\_[0] = np.mean(left\_a[-10:])  left\_fit\_[1] = np.mean(left\_b[-10:])  left\_fit\_[2] = np.mean(left\_c[-10:])  right\_fit\_[0] = np.mean(right\_a[-10:])  right\_fit\_[1] = np.mean(right\_b[-10:])  right\_fit\_[2] = np.mean(right\_c[-10:])  # x 와 y 값을 그리기 위해 생성  # 0부터 height-1(99)까지 height(100)개 만큼 1차원 배열 만들기  ploty = np.linspace(0, height-1, height)  left\_fitx = left\_fit\_[0] \* ploty \*\* 2 + left\_fit\_[1] \* ploty + left\_fit\_[2]  right\_fitx = right\_fit\_[0] \* ploty \*\* 2 + right\_fit\_[1] \* ploty + right\_fit\_[2]  img\_src[nonzero\_y[left\_lane\_inds], nonzero\_x[left\_lane\_inds]] = [255, 0, 100]  img\_src[nonzero\_y[right\_lane\_inds], nonzero\_x[right\_lane\_inds]] = [0, 100, 255]  ########################################  # 아래코드 추가 하기 : 여기  ########################################  cv2.imshow('dst',img\_src)  cv2.waitKey(0)  cv2.destroyAllWindows() |
| --- |

위의 코드에 합치기

| color\_img = np.zeros\_like(img\_src)  left = np.array([np.transpose(np.vstack([left\_fitx, ploty]))])  right = np.array([np.flipud(np.transpose(np.vstack([right\_fitx, ploty])))])  points = np.hstack((left, right))  # 차선 그리기  cv2.polylines(color\_img, np.int\_(points), False, (0, 255, 255),10)  # 차선 안쪽 채우기  #cv2.fillPoly(color\_img, np.int\_(points), (0, 255, 0))  # 원본영상과 차선검출 영상 합치기 : 가중치 조절(1:100%, 0.4:40%)  img\_src = cv2.addWeighted(img\_src, 1, color\_img, 0.4, 0) |
| --- |

예제

| import numpy as np  left\_fitx = [2,3,4]  right\_fitx = [22,23,24]  ploty = [10,11,12]  print(np.vstack([left\_fitx, ploty]))  left = np.array([np.transpose(np.vstack([left\_fitx, ploty]))])  print(left)  print(np.vstack([right\_fitx, ploty]))  right = np.array([np.flipud(np.transpose(np.vstack([right\_fitx, ploty])))])  print(right)  points = np.hstack((left, right))  print(points) |
| --- |

|  | | |
| --- | --- | --- |
| np.vstack([left\_fitx, ploty]) | np.array([np.transpose(np.vstack([left\_fitx, ploty]))]) | |

|  | | |
| --- | --- | --- |
| np.vstack([right\_fitx, ploty]) | np.array([np.flipud(np.transpose(np.vstack([right\_fitx, ploty])))]) | |

|  |
| --- |