컴퓨터공학과 201702081 최재범 5주차_과제

- 구현 코드 및 설명

◆ get_integral_image : 행렬을 탐색하며 리스트 슬라이싱으로 대상 값들 자르고, np.sum()으로 더함

```
    ################################
    # ToDo
    # dst는 integral image
    # dst 알아서 채우기
    ################################
    for integral_i in range(h): # 0 ~ h - 1
        for integral_j in range(w):
            temp = src[0:integral_i + 1, 0:integral_j + 1] # 0-0 ~ 0-h-1
            dst[integral_i][integral_j] = np.sum(temp)

    return dst
```

◆ calc_M_harris : IxIx, IxIy, IyIy 행렬을 탐색하며 M의 각 원소에 누적함

```
for row in range(h):
    for col in range(w):

        for f_row in range(fsize):
            for f_col in range(fsize):
                ################################
                # ToDo
                # 위의 2중 for문을 참고하여 M 완성
                ################################
                M[row, col, 0, 0] = M[row, col, 0, 0] + IxIx_pad[row + f_row, col + f_col]
                M[row, col, 0, 1] = M[row, col, 0, 1] + IxIy_pad[row + f_row, col + f_col]
                M[row, col, 1, 0] = M[row, col, 0, 1]
                M[row, col, 1, 1] = M[row, col, 1, 1] + IyIy_pad[row + f_row, col + f_col]

return M
```

◆ harris_detector : 공식을 이용하여 det, trace를 계산하고 Harris & Stephens 방법으로 R 계산

```
R = np.zeros((h, w))
for row in range(h):
    for col in range(w):
        ###########################################################################
        # ToDo
        # det_M 계산
        # trace_M 계산
        # R 계산 Harris & Stephens (1988), Nobel (1998) 어떤걸로 구현해도 상관없음
        ###########################################################################
        det_M = M_harris[row][col][0][0] * M_harris[row][col][1][1] - M_harris[row][col][0][1] * M_harris[row][col][1][0]
        trace_M = M_harris[row][col][0][0] + M_harris[row][col][1][1]
        R[row, col] = det_M - (0.04 * (trace_M ** 2))
```

◆ harris_detector_integral : 구한 Integral에서 필터 길이만큼 먼 위치의 값을 이용하여 np.sum을 이용한 것처럼 누적한 효과를 냄

```
##############################
# ToDo
# M_integral 완성시키기
##############################
M_integral = np.zeros((h, w, 2, 2))
for row in range(5, h):
    for col in range(5, w):
        # D-B-C+A : np.sum 대신함
        M_integral[row, col, 0, 0] = IxIx_integral[row, col] - IxIx_integral[row - fsize, col] - IxIx_integral[row, col - fsize] + IxIx_integral[row - fsize, col - fsize]
        M_integral[row, col, 1, 1] = IyIy_integral[row, col] - IyIy_integral[row - fsize, col] - IyIy_integral[row, col - fsize] + IyIy_integral[row - fsize, col - fsize]
        M_integral[row, col, 0, 1] = IxIy_integral[row, col] - IxIy_integral[row - fsize, col] - IxIy_integral[row, col - fsize] + IxIy_integral[row - fsize, col - fsize]
        M_integral[row, col, 1, 0] = M_integral[row, col, 0, 1]
```

또한 harris_detector 함수에서 썼던 방법과 동일하게 det, trace, R을 구함

```
R = np.zeros((h, w))
for row in range(h):
    for col in range(w):
        ###########################################################################
        # ToDo
        # det_M 계산
        # trace_M 계산
        # R 계산 Harris & Stephens (1988), Nobel (1998) 어떤걸로 구현해도 상관없음
        ###########################################################################
        det_M = M_integral[row][col][0][0] * M_integral[row][col][1][1] - M_integral[row][col][0][1] * M_integral[row][col][1][0]
        trace_M = M_integral[row][col][0][0] + M_integral[row][col][1][1]
        R[row, col] = det_M - (0.04 * (trace_M ** 2))
```

● 이미지

◆ Original

◆ Harris

◆ Harris Integral

◆ 소요 시간

```
C:\Users\l4m6d4\anaconda3\python.exe C:/Users/l4m6d4/Desktop/integ/integral_image_report.py
start!
M_harris time :  13.716426341
make integral image time :  28.802940863999996
M_harris integral time :  0.9178834269999996

Process finished with exit code 0
```

Integral을 구하는데 시간이 오래 걸리긴 하지만, 일단 이것을 구해 놓으면 계산이 확실히 빠르게 되는 것을 확인하였다.

- 느낀 점

이론이 이해가 안가서 많이 헤맸다. 힘들다

- 과제 난이도

막 쉽지는 않은데 다 하고 보니 엄청 어렵지도 않은 것 같다

주석으로 힌트를 얻은 것이 많은 도움이 되었다. 없으면 엄청 오래 걸렸을 것 같다