이름 : 강시온

학번 : 201802045

학과 : 컴퓨터공학과

구현 코드

```python
def get_integral_image(src):
    ###############################################################################
    # ToDo
    # src를 integral로 변경하는 함수
    # 실습 때 진행한 내용입니다.
    ###############################################################################
    assert len(src.shape) == 2
    h, w = src.shape
    dst = np.zeros(src.shape)
    for row in range(h):
        dst[row, 0] = np.sum(src[0:row+1, 0])

    for col in range(w):
        dst[0, col] = np.sum(src[0, 0:col+1])

    for row in range(1, h):
        for col in range(1, w):
            dst[row, col] = src[row, col] + dst[row-1, col] + dst[row, col-1] - dst[row-1, col-1]

    return dst
def calc_local_integral_value(src, left_top, right_bottom):
    ###############################################################################
    # ToDo
    # integral에서 filter의 요소합 구하기
    # 실습 때 진행한 내용입니다.
    ###############################################################################
    assert len(left_top) == 2
    assert len(right_bottom) == 2

    lt_row, lt_col = left_top
    rb_row, rb_col = right_bottom

    lt_val = src[lt_row - 1, lt_col - 1]
    rt_val = src[lt_row - 1, rb_col]
    lb_val = src[rb_row, lt_col - 1]
    rb_val = src[rb_row, rb_col]

    if lt_row == 0:
        lt_val = 0
        rt_val = 0
    if lt_col == 0:
        lt_val = 0
        lb_val = 0

    return lt_val - lb_val - rt_val + rb_val
```

```
##########################################################################
# ToDo
# integral을 사용하지 않고 covariance matrix 구하기
# 4중 for문을 채워서 완성하기
# 실습 시간에 했던 내용을 생각하면 금방 해결할 수 있음
##########################################################################
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] += IxIx_pad[row+f_row, col+f_col]
                M[row, col, 0, 1] += IxIy_pad[row+f_row, col+f_col]
                M[row, col, 1, 0] += IxIy_pad[row+f_row, col+f_col]
                M[row, col, 1, 1] += IyIy_pad[row+f_row, col+f_col]
```

```
##########################################################################
# ToDo
# det_M 계산
# trace_M 계산
# R 계산 Harris 방정식 구현
##########################################################################
det_M = M_harris[row, col, 0, 0] * M_harris[row, col, 1, 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 - k * (trace_M * trace_M)
```

```
##########################################################################
# ToDo
# integral 값을 이용하여 covariance matrix 구하기
# calc_local_integral_value를 사용하면 쉽게 구할 수 있음
##########################################################################
for row in range(h):
    for col in range(w):
        left_top = (row, col)
        right_bottom = (row + fsize - 1, col + fsize - 1)
        M[row, col, 0, 0] = calc_local_integral_value(IxIx_integral_pad, left_top, right_bottom)
        M[row, col, 0, 1] = calc_local_integral_value(IxIy_integral_pad, left_top, right_bottom)
        M[row, col, 1, 0] = M[row, col, 0, 1]
        M[row, col, 1, 1] = calc_local_integral_value(IyIy_integral_pad, left_top, right_bottom)

return M
```

코드 설명

Calc_integral_image, calc_local_integral_value 해당 두 함수는 실습에서 진행한 코드를 그대로 사용하였다.

누적합을 이용하여 현재 부분의 사각형 내부의 숫자의 합을 구하는 코드이다.

Integral을 사용하지않고 covariance matrix를 구하는 3번째 사진에 해당하는 코드는 주석으로 작성된 코드를 4중포문으로 작성하기위해 np.sum으로 row:row+fsize, col:col+fsize로 범위로 지정된 배열의 부분을 2중포문으로 직접 순차탐색하면서 합을 쌓아주는식으로 구현하였습니다.

4번째 사진인 det_M, trace_M은 저번시간에 구한

$$har = \det[\mu(\sigma_I, \sigma_D)] - \alpha[\text{trace}(\mu(\sigma_I, \sigma_D))^2] =$$
$$g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

해당 해리스 방정식을 통하여 IxIx는 0,0 IxIy는 0,1 IyIy는 1,1에 좌표에 담겨있는 값을 활용하여 구하였다.

Integral을 이용하여 covariance matrix를 구하는 코드인 5번째 사진은 실습시간에 구현한 Calc_integral_image, calc_local_integral_value 두함수를 이용해 (row, col) 부터 (row+fsize-1, col+fsize-1)까지의 사각형의 합을 구해주었다.

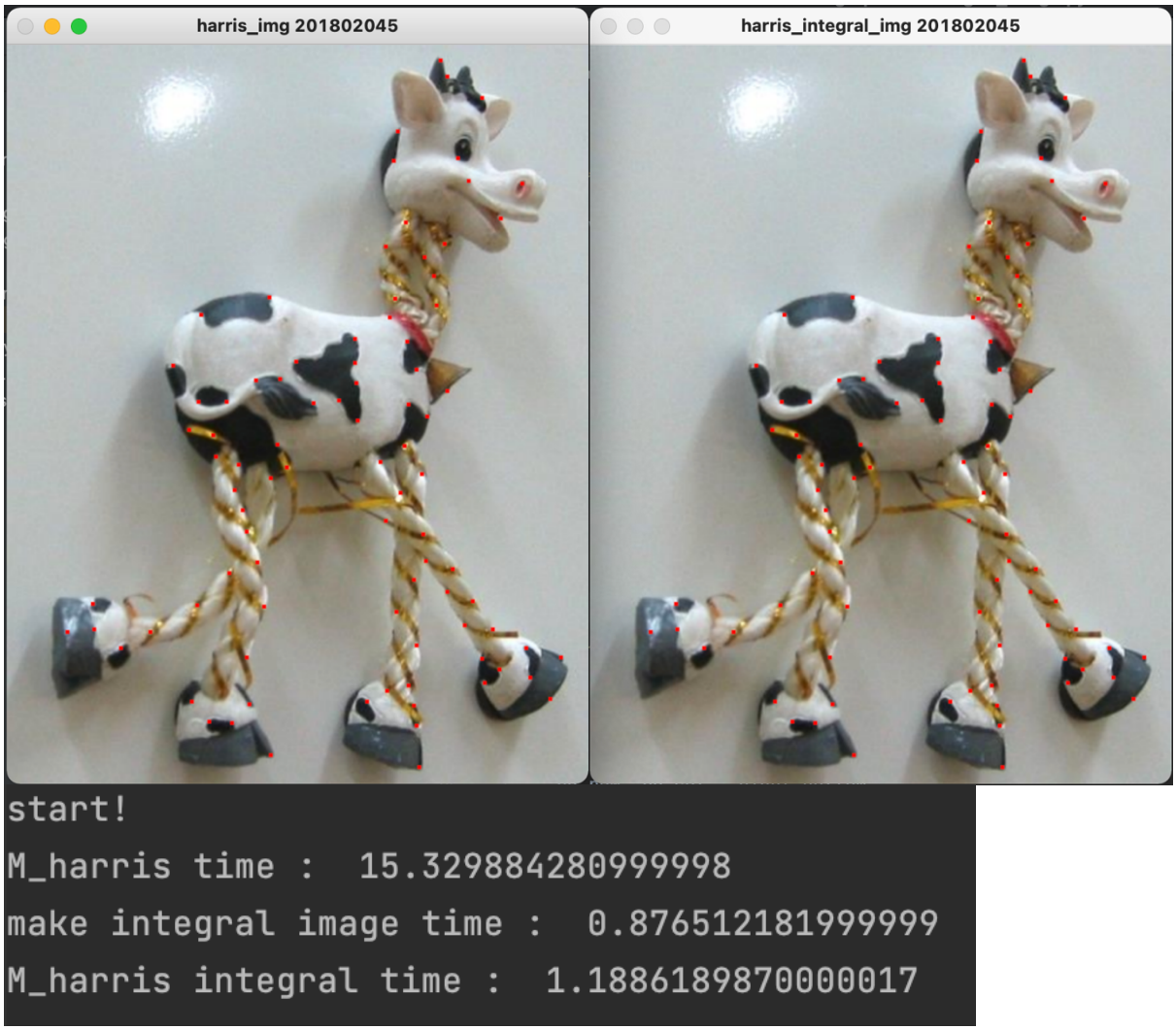
현재 fsize가 5이므로 552 x 435사이즈의 해당이미지에 필터링을 직접 적용할경우 5x5의 필터를 552 x 435의 각 칸에 모두 적용하므로 6003000 횟수의 연산을 수행해야 한다.

하지만 Integral을 활용한다면 3번의 연산만들 통해 5x5필터를 한번에 적용한 합을 구할 수 있으므로 720360 횟수의 연산만을 수행하면 된다.

그래서 실제로 아래 결과 이미지를 보면

Integral을 사용한 harris_detection이 훨신 빠른 수행속도를 보여주는것을 알 수 있다

결과 이미지



난이도

난이도는 이론시간에 배운것들을 정확하게 활용해 볼 수 있는 적절한 난이도 였던것 같습니다.