컴퓨터 그래픽스 과제 #03

학과 : 컴퓨터공학과
학번 : 201802161
이름 : 조은빈

1. 구현 코드

* calcMagnitude

```
55    # Ix와 Iy의 magnitude를 구함
56    def calcMagnitude(Ix, Iy):
57        ########################################
58        # TODO                                 #
59        # calcMagnitude 완성                   #
60        # magnitude : ix와 iy의 magnitude      #
61        ########################################
62        # Ix와 Iy의 magnitude를 계산
63        magnitude = np.sqrt(Ix**2 + Iy**2)
64        return magnitude
65
```

* calcAngle

```
67    # Ix와 Iy의 angle을 구함
68    def calcAngle(Ix, Iy):
69        ######################################
70        # TODO                               #
71        # calcAngle 완성                     #
72        # angle     : ix와 iy의 angle        #
73        ######################################
74        angle = np.arctan(Iy/Ix)
75        angle = np.rad2deg(angle)
76        return angle
```

* non_maximum_supression

```python
# non-maximum supression 수행
def non_maximum_supression(magnitude, angle):
    ###############################################################################
    # TODO                                                                        #
    # non_maximum_supression 완성                                                  #
    # larger_magnitude     : non_maximum_supression 결과(가장 강한 edge만 남김)    #
    ###############################################################################
    (h, w) = magnitude.shape
    cv2.imshow('before non maximum supression', magnitude/255)
    larger_magnitude = np.zeros((h, w))
    for row in range(1, h - 1):
        for col in range(1, w - 1):
            degree = angle[row, col]

            # gradient의 degree는 edge와 수직방향이다.
            if 0 <= degree and degree < 45:
                rate = np.tan(np.deg2rad(degree))
                left_magnitude = (rate) * magnitude[row - 1, col - 1] + (1 - rate) * magnitude[row, col - 1]
                right_magnitude = (rate) * magnitude[row + 1, col + 1] + (1 - rate) * magnitude[row, col + 1]
                if magnitude[row, col] == max(left_magnitude, magnitude[row, col], right_magnitude):
                    larger_magnitude[row, col] = magnitude[row, col]

            elif -45 > degree and degree >= -90:
                rate = 1/np.tan(np.deg2rad(degree))
                up_magnitude = (rate) * magnitude[row - 1, col + 1] + (1 - rate) * magnitude[row - 1, col]
                down_magnitude = (rate) * magnitude[row + 1, col - 1] + (1 - rate) * magnitude[row + 1, col]
                if magnitude[row, col] == max(up_magnitude, magnitude[row, col], down_magnitude):
                    larger_magnitude[row, col] = magnitude[row, col]

            elif -45 <= degree and degree < 0:
                rate = np.tan(np.deg2rad(degree))
                left_magnitude = (rate) * magnitude[row + 1, col - 1] + (1 - rate) * magnitude[row, col - 1]
                right_magnitude = (rate) * magnitude[row - 1, col + 1] + (1 - rate) * magnitude[row, col + 1]
                if magnitude[row, col] == max(left_magnitude, magnitude[row, col], right_magnitude):
                    larger_magnitude[row, col] = magnitude[row, col]

            elif 90 >= degree and degree >= 45:
                rate = 1/np.tan(np.deg2rad(degree))
                up_magnitude = (rate) * magnitude[row - 1, col - 1] + (1 - rate) * magnitude[row - 1, col]
                down_magnitude = (rate) * magnitude[row + 1, col + 1] + (1 - rate) * magnitude[row + 1, col]
                if magnitude[row, col] == max(up_magnitude, magnitude[row, col], down_magnitude):
                    larger_magnitude[row, col] = magnitude[row, col]

            else:
                # angle을 np.arctna(Iy/Ix) 로 구했는데 Ix값이 0일 경우 해당 angle은 nan값이 저장됨
                print(row, col, 'error!  degree :', degree)

    larger_magnitude = (larger_magnitude / np.max(larger_magnitude) * 255).astype(np.uint8)
    cv2.imshow('after non maximum supression', larger_magnitude)
    return larger_magnitude
```

* double_thresholding

```python
# double_thresholding 수행 high threshold value는 내장함수(otsu방식 이용)를 사용하여 구하고
def double_thresholding(src, test_mode=False):
    (h, w) = src.shape
    high_threshold_value, _ = cv2.threshold(src, 0, 255, cv2.THRESH_OTSU)
    print('highthreshold')
    print(high_threshold_value)
    if test_mode == True:
        print('test mode!! - double threshold function')
        high_threshold_value = 200
    low_threshold_value = high_threshold_value * 0.4
    weak_edge = low_threshold_value + 1
    dst = src.copy()
    for row in range(h):
        for col in range(w):
            if dst[row, col] >= high_threshold_value:
                dst[row, col] = 255
            elif dst[row, col] < low_threshold_value:
                dst[row, col] = 0
            else:
                ################################################
                # TODO                                         #
                # high 보다는 작고 low보다는 큰 경우             #
                ################################################
                dst[row, col] = weak_edge

    all_strong = np.sum(dst == 255)
    while(1):
        find_strong(dst, h, w, weak_edge)
        if all_strong == np.sum(dst == 255):
            break;
        all_strong = np.sum(dst == 255)

    for row in range(1, h-1):
        for col in range(1, w-1):
            if(dst[row, col] == weak_edge):
                dst[row, col] = 0
    return dst
```

```python
def find_strong(dst, h, w, weak_edge):
    for row in range(1, h-1):
        for col in range(1, w-1):
            if dst[row, col] == weak_edge:
                if dst[row+1, col+1] == 255 or dst[row+1, col-1] == 255 or dst[row+1, col] == 255 or dst[row, col+1] == 255 \
                        or dst[row, col-1] == 255 or dst[row-1, col-1] == 255 or dst[row-1, col+1] == 255 or dst[row-1, col] == 255:
                    dst[row, col] = 255
```

2. 코드 설명

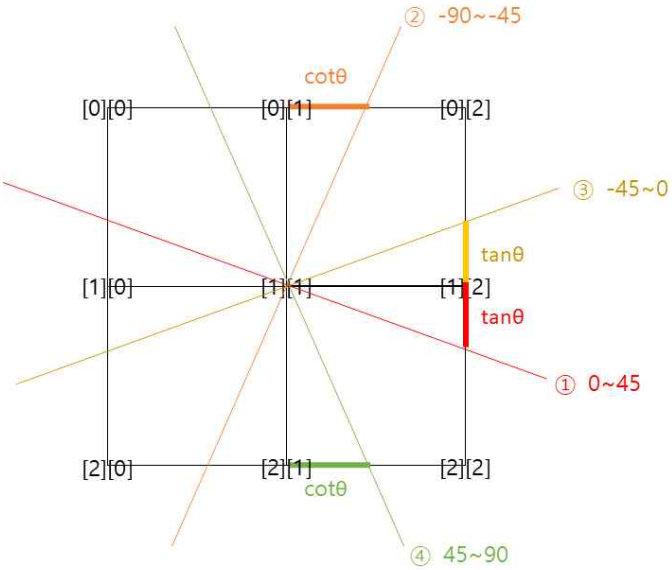
* calcMagnitude

$\sqrt{I_x^2 + I_y^2}$ 을 코드로 구현한다.

* calcAngle

$\tan^{-1}(\frac{I_y}{I_x})$을 코드로 구현한다.

* non_maximum_supression



각각의 각도에 따라 rate를 다르게 구해준다.

* double_thresholding

weak edge 값을 지정하고 heigh 보다는 작고 low보다는 큰 경우 weak edge 값을 저장한다. weak edge인 픽셀 주위의 8픽셀 중 strong edge가 있으면 weak edge를 strong edge로 확정한다. strong edge의 수가 변하지 않을 때까지 반복한다. weak edge를 strong edge로 확정한 후 남아있는 weak edge는 edge가 아니므로 0을 저장한다.

3. 이미지

* original

* my canny edge detection

* lx

* ly

* before non maximum supression

* after non maximum supression

4. 느낀 점

결과 이미지가 다르게 나오더라도 어디에서 잘못된 건지 찾기가 어려웠다. 저번 과제에서는 가우시안 식 결과값으로 내가 구현한 코드가 맞는지 판단할 수 있었지만 이번 과제에서는 내가 구현한 코드가 맞는지 판단할 수가 없어서 난감했다.

이론 수업에서는 Non-maximum suppression 파트에서 두 픽셀 간의 거리에 따라 가중치를 달리해서 magnitude를 구한다는 개념에 대해서만 배웠다. 과제 설명 동영상으로 가중치를 계산하고 적용하는 방법에 대해 설명을 들었지만 이해가 잘 안돼서 코드로 구현하는데 많이 헤맸다.

5. 과제 난이도

저번 과제보다 어려웠다.