# 영상처리 실습 보고서

## 7주차: Canny edge detection

| | |
|---|---|
| 학번 | 201802170 |
| 이름 | 하 상 호 |

## 1. 과제의 내용

- Canny edge detection 구현하기
  low pass 필터 적용하기(ex: Gaussian filter)
  high pass 필터 적용하기(ex: sobel filter)
  magnitude와 angle 구하기
  non-max suppression 수행
  double thresholding 수행

## 2. 과제의 해결 방법

과제의 내용을 해결하기 위해 어떠한 방법을 사용했는지, 자세하게 기술한다.

- 6개의 함수를 구현해서 canny edge detection을 완성
- apply_gaussian_filter
- apply_sobel_filter
- calc_magnitude
- calc_angle
- non_maximum_suppression
- double_thresholding

```python
def apply_gaussian_filter(src, fsize=3, sigma=1):
    ###########################################################
    # TODO                                                    #
    # src에 gaussian filter 적용                               #
    ###########################################################
    dst = my_filtering(src, get_Gaussian_mask(fsize, sigma))

    return dst

def apply_sobel_filter(src):
    ###########################################################
    # TODO                                                    #
    # src에 sobel filter 적용                                  #
    ###########################################################
    x, y = get_sobel_mask()
    Ix, Iy = my_filtering(src, x), my_filtering(src, y)
    return Ix, Iy
```

low, high pass 필터를 적용하기 위해  apply_gaussian_filter , apply_sobel_filter 함수를 구현

본 함수는 이전 과제, 실습들을 참고하여 구현하였다.
과제 파일에 각각의 mask 작업이 작성되어 있으므로, mask 작업을 이용하여
src : img를 받아 filtering을 통해 low , high pass 필터를 적용한다.

```python
def calc_magnitude(Ix, Iy):
    ###########################################################
    # TODO                                                    #
    # Ix, Iy 로부터 magnitude 계산                             #
    ###########################################################

    magnitude = np.sqrt(Ix ** 2 + Iy ** 2)

    return magnitude


def calc_angle(Ix, Iy, eps=1e-6):
    ###########################################################
    # TODO                                                    #
    # Ix, Iy 로부터 angle 계산                                 #
    # numpy의 arctan 사용 O, arctan2 사용 X                    #
    # 호도법이나 육십분법이나 상관 X                            #
    # eps     : Divide by zero 방지용                          #
    ###########################################################
    angle = np.arctan(Iy / (Ix + eps))  # 0으로 나눠지는거 방지
    return np.rad2deg(angle)
```

magnitude와 angle 구하는 방법은 간단하다.

magnitude 는 Ix, Iy를 받은 것을 np.sprt (루트) 를 이용하여 Ix ^2 + Iy^2를 사용한다.

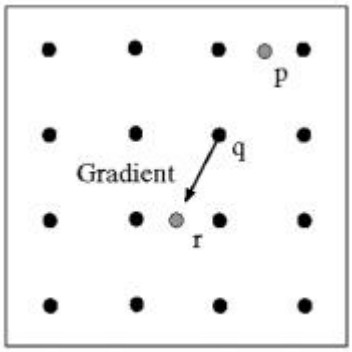
angle 의 경우 eps 값을 갖는데 0으로 나누어지는 것을 방지하기 위해 사용한다. angle 값은 np.arctan를 이용한다. return 은 rad2deg를 통해 angle을 반환한다.

```
def non_maximum_supression(magnitude, angle):
    ###############################################################################################
    # TODO                                                                                        #
    # Non-maximum-supression 수행                                                                  #
    # 스켈레톤 코드는 angle이 육십분법으로 나타나져 있을 것으로 가정                                    #
    ###############################################################################################
    (h, w) = magnitude.shape
    # angle의 범위 : -90 ~ 90
    largest_magnitude = np.zeros((h, w))
    for row in range(1, h - 1):
        for col in range(1, w - 1):
            degree = angle[row, col]
```

```
        ratian = np.deg2rad(degree)
        if 0 <= degree and degree < 45:
            rat = np.tan(ratian)
            p = magnitude[row, col + 1] * (1 - rat) + magnitude[row + 1, col + 1] * rat
            r = magnitude[row, col - 1] * (1 - rat) + magnitude[row - 1, col - 1] * rat

        elif 45 <= degree and degree <= 90:
            rat = np.tan(np.pi/2-ratian)
            p = magnitude[row + 1, col] * (1 - rat) + magnitude[row + 1, col + 1] * rat
            r = magnitude[row - 1, col] * (1 - rat) + magnitude[row - 1, col - 1] * rat
        elif -45 <= degree and degree < 0:
            rat = np.tan(-ratian)
            p = magnitude[row, col + 1] * (1 - rat) + magnitude[row - 1, col + 1] * rat
            r = magnitude[row, col - 1] * (1 - rat) + magnitude[row + 1, col - 1] * rat
        elif -90 <= degree and degree < -45:
            rat = np.tan(ratian + np.pi/2)
            p = magnitude[row - 1, col] * (1 - rat) + magnitude[row - 1, col + 1] * rat
            r = magnitude[row + 1, col] * (1 - rat) + magnitude[row + 1, col - 1] * rat
        else:
            print(row, col, 'error!  degree :', degree)
        if magnitude[row, col] == max(p, r, magnitude[row, col]):
            largest_magnitude[row, col] = magnitude[row, col]
    return largest_magnitude
```

non_maximum_suppression 의 경우 본 함수가 왜 , 어떻게 실행 되는지 알아야 한다.




q 값이 주어 졌을 때 각도에 맞는 선분을 그어놓는데 p 만이 아니라 바로 180도 반대쪽 r 값에도 접근을 해야 한다.

본 함수는 magnitude, angle 값을 인자로 받는데 받은 인자를 통해서 함수를 수행 한다.
if 문을 통해 degree 값이 0 ~ 45 , 45 ~ 90 , -45 ~ 0, -90 ~ -45 의 경우를 나누고 p 의 값, r 의 값을 접근한다.

위 과정을 통해 Non-maximum-supression을 수행한다.

```
dst = src.copy()

# dst 범위 조정 0 ~ 255
dst = (dst - np.min(dst)) / (np.max(dst) - np.min(dst))
dst *= 255
dst = dst.astype('uint8')

# threshold는 정해진 값을 사용
high_threshold_value = 40
low_threshold_value = 5

print(high_threshold_value, low_threshold_value)
```

```
#####################################################
# TODO                                              #
# Double thresholding 수행                          #
#####################################################
(h, w) = dst.shape
# 강한 엣지는 255로, 약한 엣지는 100으로 설정.
for row in range(h):
    for col in range(w):
        if dst[row][col] >= int(high_threshold_value):
            dst[row][col] = 255
        elif dst[row][col] < int(low_threshold_value):
            dst[row][col] = 0
        else:
            dst[row][col] = 100
```

```
for row in range(h):
    for col in range(w):
        if dst[row][col] == 255:
            mid = deque()
            mid.append((row, col))
            while mid:  # BFS 진행
                x, y = mid.popleft()
                dx = [-1, -1, -1, 0, 0, 1, 1, 1]
                dy = [-1, 0, 1, -1, 1, -1, 0, 1]
                for k in range(8):
                    nx = x + dx[k]
                    ny = y + dy[k]
                    if nx >= 0 and nx < h and ny >= 0 and ny < w and dst[nx][ny] == 100:  # 약한 엣지인 경우
                        mid.append((nx, ny))
                        dst[nx][ny] = 255  # 강한 엣지로 만들어 준다.

    # 강한 엣지로 변환되지 않은 약한 엣지는 0으로.
for row in range(h):
    for col in range(w):
        if dst[row][col] != 255:
            dst[row][col] = 0

# return dst
dst = dst.astype('float32') / 255.0
return dst
```

dst 의 min max 정규화를 통해 0 ~ 225 값으로 조정 한뒤
정해진 threshold 값을 사용한다.

먼저 강한 엣지는 255 로 약한 엣지는 100으로 설정한다.
int threshold value 값을 넘어가면 255 , low threshold value 값 보다 작으면
0 값으로 나머진 100 으로 바꾼다.

bfs를 이용하여 이미지처리를 진행한다. deque를 이용하여 선들을
이어주는 작업을 수행한다.

강한 엣지로 변환되지 않은 약한 엣지는 0으로 변환 해주는 작업을 수행

double thresholding이 완료된다.

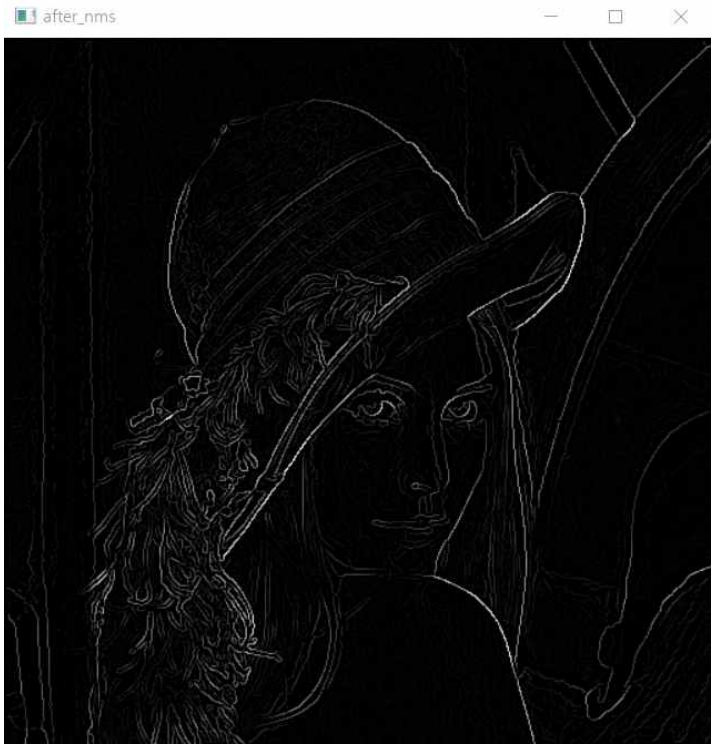
## 3. 결과물

결과물이 잘 보이도록 화면을 캡처해 보고서에 올린다.

original


magnitude

after_nms




canny_edge