Image Processing

실습 6.

2021. 04. 18.

실습 수업 소개

- **과목 홈페이지**
  - 충남대학교 사이버 캠퍼스 (http://e-learn.cnu.ac.kr)

- **TA 연락처**
  - 신준호
  - wnsgh578@naver.com

- **튜터 연락처**
  - 한승오
  - so.h4ns@gmail.com

- **실습 중 질문사항**
  - 실시간 수업중 질문 or 메일을 통한 질문
  - 메일로 질문할 때 [IP] 를 제목에 붙여주세요

실습 수업 소개

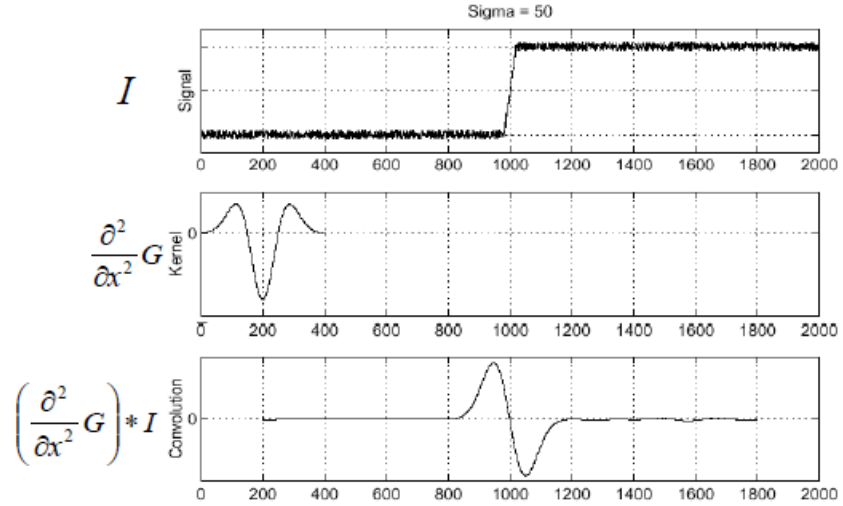
- **실습 출석**
  - 사이버캠퍼스를 통해 Zoom 출석
  - Zoom 퇴장 전 채팅 기록[학번 이름] 남기고 퇴장
  - 위 두 기록을 통해 출석 체크 진행 예정

목 차

Laplacian of Gaussian(LoG)

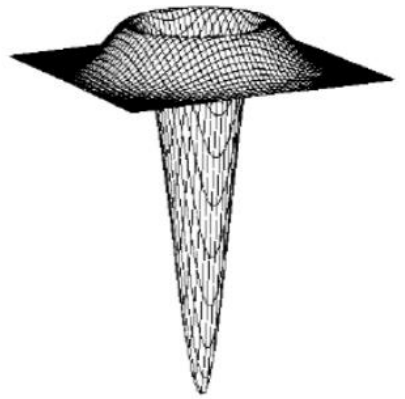
- **LoG**
  - Gaussian 필터로 noise를 감소시키고 laplacian 필터를 적용
  - Gaussian 식에 미분을 2번 하면 절차 간소화
  - zero – crossing point

- **In 1D, consider** $\frac{\partial^2}{\partial x^2}(G * I) = \left(\frac{\partial^2}{\partial x^2} G\right) * I$



- **Edge is the zero-crossing of the bottom graph**

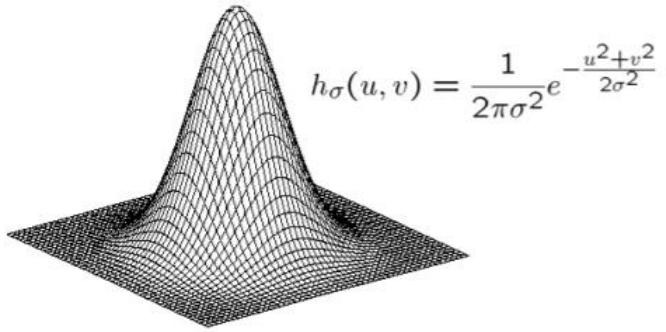
$\nabla^2 h_\sigma(u, v)$



Laplacian of Gaussian

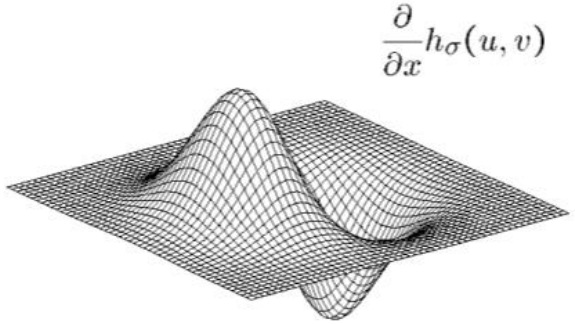
Laplacian of Gaussian(LoG)

- 여러가지 필터 모양



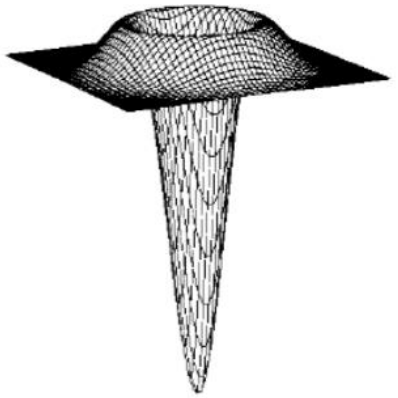
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

Gaussian



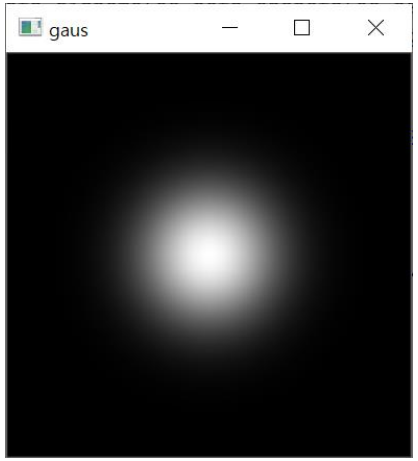
$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

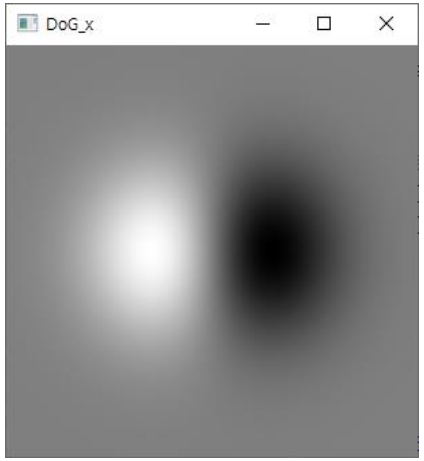
Derivative of Gaussian

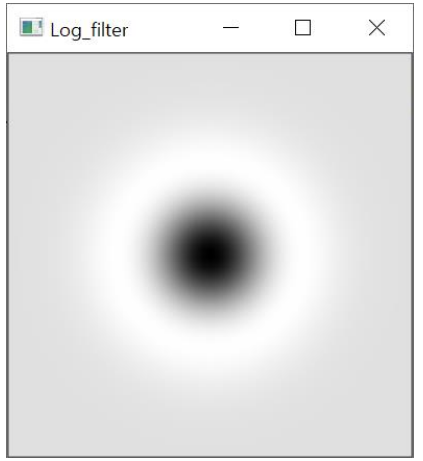


$$\nabla^2 h_\sigma(u,v)$$

Laplacian of Gaussian







Laplacian of Gaussian(LoG)

- LoG 실습

```python
def get_LoG_filter(fsize=3, sigma=1):
    y, x = np.mgrid[-(fsize // 2):(fsize // 2) + 1, -(fsize // 2):(fsize // 2) + 1]

    r_square = x**2 + y**2

    LoG_filter = (r_square / sigma**4) - (2 / sigma**2)
    LoG_filter = LoG_filter * np.exp(-r_square / (2 * sigma**2))

    LoG_filter = LoG_filter - (LoG_filter.sum() / fsize**2)

    return LoG_filter
```

$$G(x,y) = \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

$$\nabla^2 G(x,y) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

$$= \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2}\right)\exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

$$= \left(\frac{r^2}{\sigma^4} - \frac{2}{\sigma^2}\right)\exp\left(-\frac{r^2}{2\sigma^2}\right)$$

Laplacian of Gaussian(LoG)

- LoG 실습

```
[[ 0.     0.     0.001  0.003  0.005  0.003  0.001  0.     0.   ]
 [ 0.     0.002  0.017  0.054  0.078  0.054  0.017  0.002  0.   ]
 [ 0.001  0.017  0.11   0.246  0.271  0.246  0.11   0.017  0.001]
 [ 0.003  0.054  0.246  0.    -0.607  0.     0.246  0.054  0.003]
 [ 0.005  0.078  0.271 -0.607 -2.    -0.607  0.271  0.078  0.005]
 [ 0.003  0.054  0.246  0.    -0.607  0.     0.246  0.054  0.003]
 [ 0.001  0.017  0.11   0.246  0.271  0.246  0.11   0.017  0.001]
 [ 0.     0.002  0.017  0.054  0.078  0.054  0.017  0.002  0.   ]
 [ 0.     0.     0.001  0.003  0.005  0.003  0.001  0.     0.   ]]
```

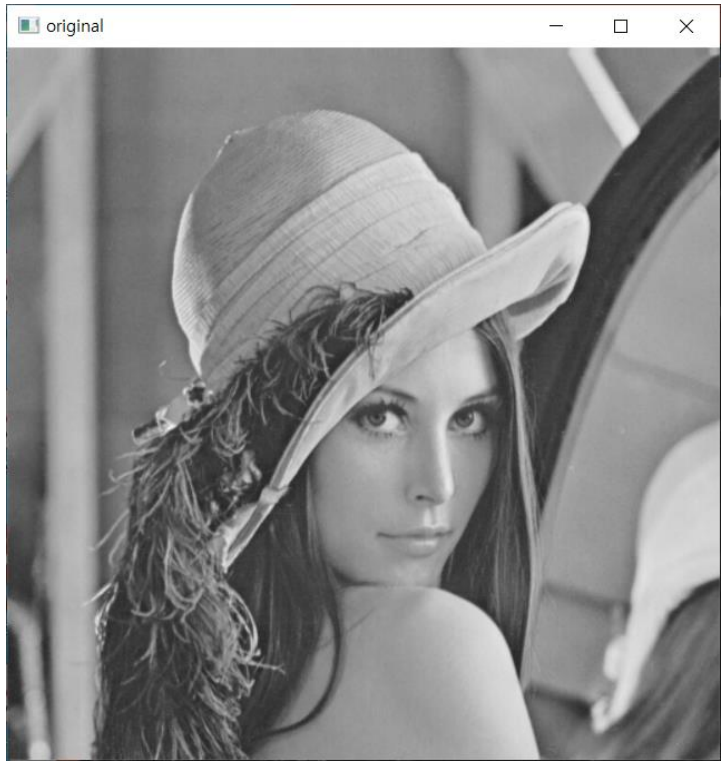
```python
if __name__ == '__main__':
    img = cv2.imread('Lenna.png', cv2.IMREAD_GRAYSCALE)

    Log_filter = get_LoG_filter(fsize=9, sigma=1)

    with np.printoptions(precision=3, suppress=True):
        print(Log_filter)

    dst = my_filtering(img, Log_filter, 'repetition')

    dst = np.sqrt(dst**2)
    dst = dst - dst.min()
    dst = dst / dst.max()

    cv2.imshow('LoG', dst)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Laplacian of Gaussian(LoG)

- **LoG 실습**

과제

- **Canny edge detection 구현하기**



original

Canny Edge Detection

과제

- **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

과제

- **Canny edge detection 구현하기**
  - ➢ low pass 필터 적용하기
  - ➢ high pass 필터 적용하기
  - ➢ magnitude와 angle 구하기
  - ➢ non-max suppression 수행
  - ➢ double thresholding 수행

```python
def canny_edge_detection(src):
    # Apply low pass filter
    I = apply_gaussian_filter(src, fsize=3, sigma=1)

    # Apply high pass filter
    Ix, Iy = apply_sobel_filter(I)

    # Get magnitude and angle
    magnitude = calc_magnitude(Ix, Iy)
    angle = calc_angle(Ix, Iy)

    # Apply non-maximum-supression
    after_nms = non_maximum_supression(magnitude, angle)

    # Apply double thresholding
    dst = double_thresholding(after_nms)

    return dst, after_nms, magnitude
```

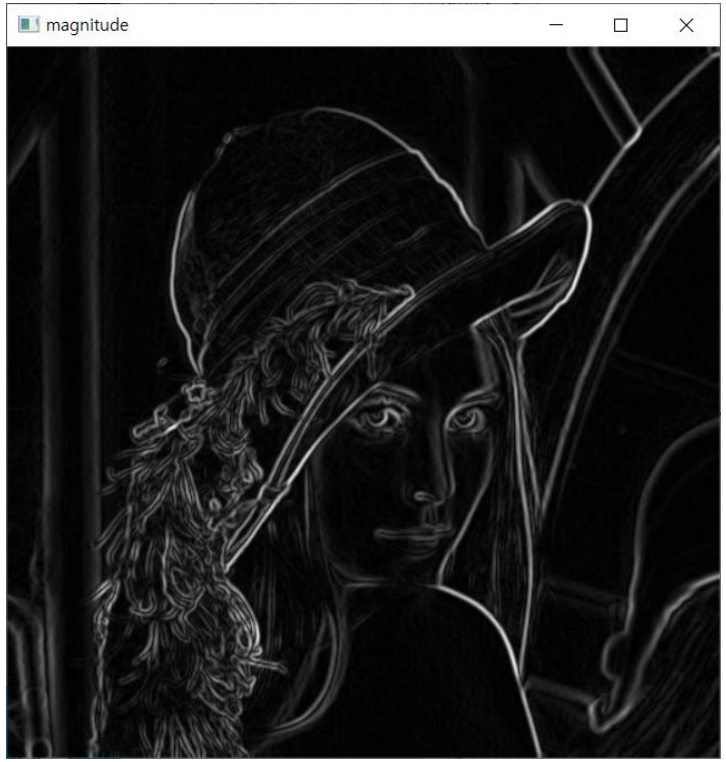
과제

- **Canny edge detection 구현하기**
  - ➢ low pass 필터 적용하기
  - ➢ high pass 필터 적용하기

```python
def apply_gaussian_filter(src, fsize=3, sigma=1):
    ###########################################################
    # TODO                                                    #
    # src에 gaussian filter 적용                               #
    ###########################################################
    dst = None

    return dst


def apply_sobel_filter(src):
    ###########################################################
    # TODO                                                    #
    # src에 sobel filter 적용                                  #
    ###########################################################
    Ix, Iy = None, None

    return Ix, Iy
```

과제

- **Canny edge detection 구현하기**
  - ➢ magnitude 계산




```python
def calc_magnitude(Ix, Iy):
    ##########################################################
    # TODO                                                   #
    # Ix, Iy 로부터 magnitude 계산                            #
    ##########################################################
    magnitude = None

    return magnitude
```

과제

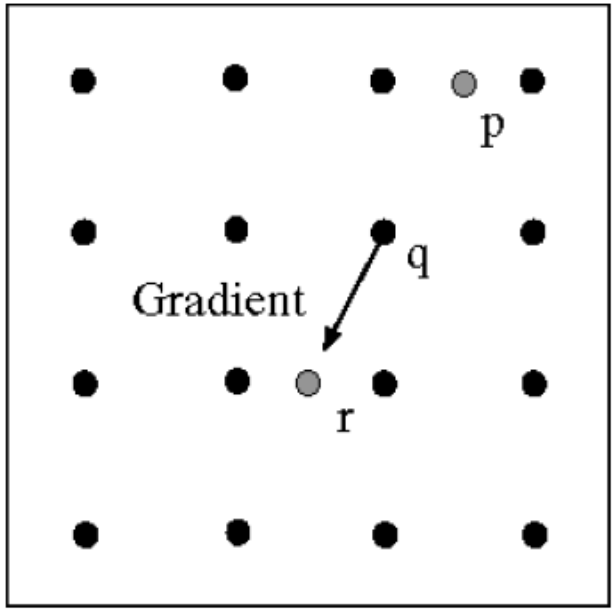
- **Canny edge detection 구현하기**
  - ➢ angle 계산(호도법, 육십분법 중 편한 것을 사용해서 구현)
  - ➢ np.arctan과 np.arctan2는 출력값 범위가 다름. np.arctan 사용

```python
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 = None

    return angle
```

np.rad2deg(radian) = degree
np.deg2rad(degree) = radian

과제

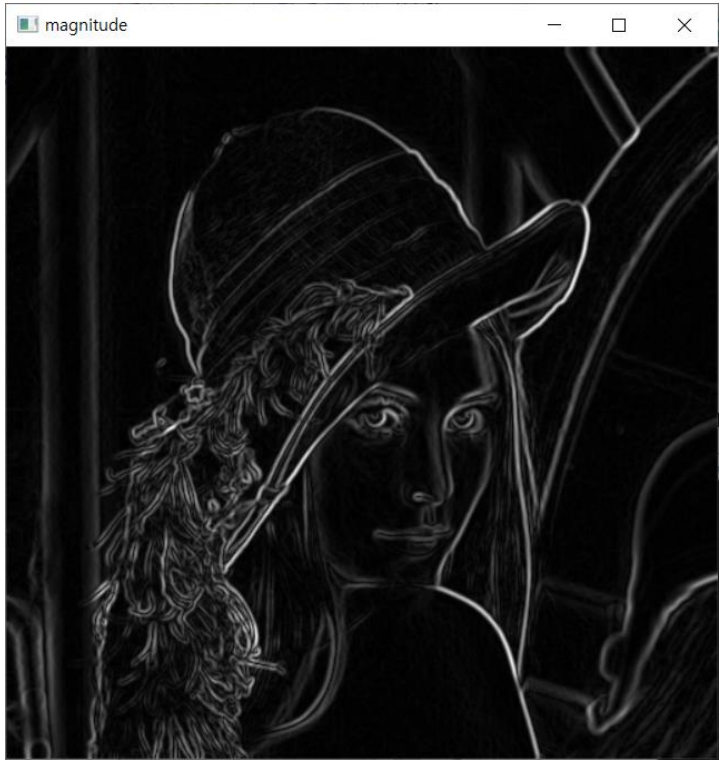
- **Canny edge detection 구현하기**
  - ➢ non-max suppression 수행



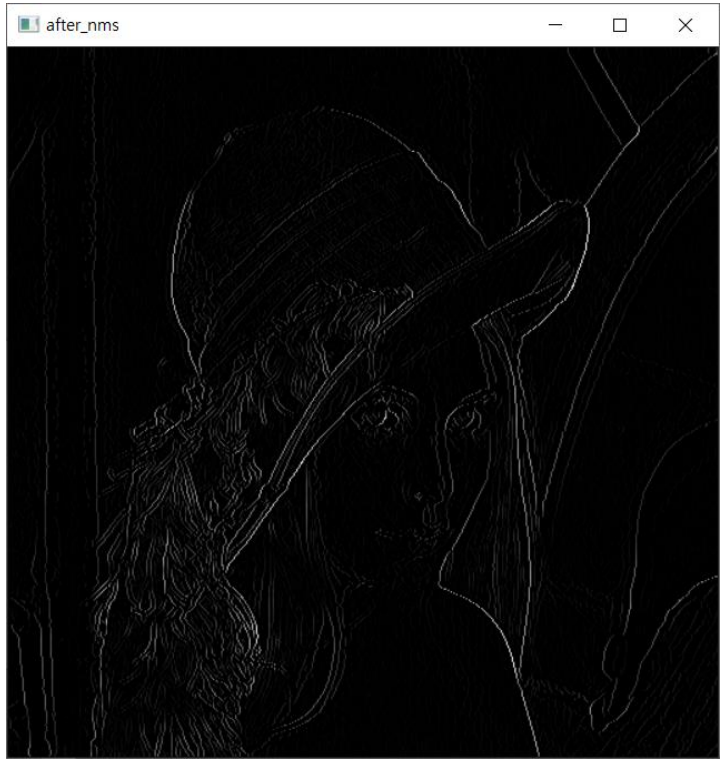

```python
def non_maximum_supression(magnitude, angle):
    ##########################################################################################
    # TODO                                                                                   #
    # Non-maximum-supression 수행                                                             #
    # 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]

            # 각도가 d일 때
            # d 각도의 픽셀과 동시에 180 + d 각도 방향의 픽셀과도 비교 해야함.
            # ex) 10도와 190도 -> 대략 우측과 좌측 픽셀
            # interpolation 방법은 linear로 구현
            if 0 <= degree and degree < 45:
                pass
            elif 45 <= degree and degree <= 90:
                pass
            elif -45 <= degree and degree < 0:
                pass
            elif -90 <= degree and degree < -45:
                pass
            else:
                print(row, col, 'error!  degree :', degree)

    return largest_magnitude
```

과제

- **Canny edge detection 구현하기**
  - ➢ non-max suppression 수행



before



after

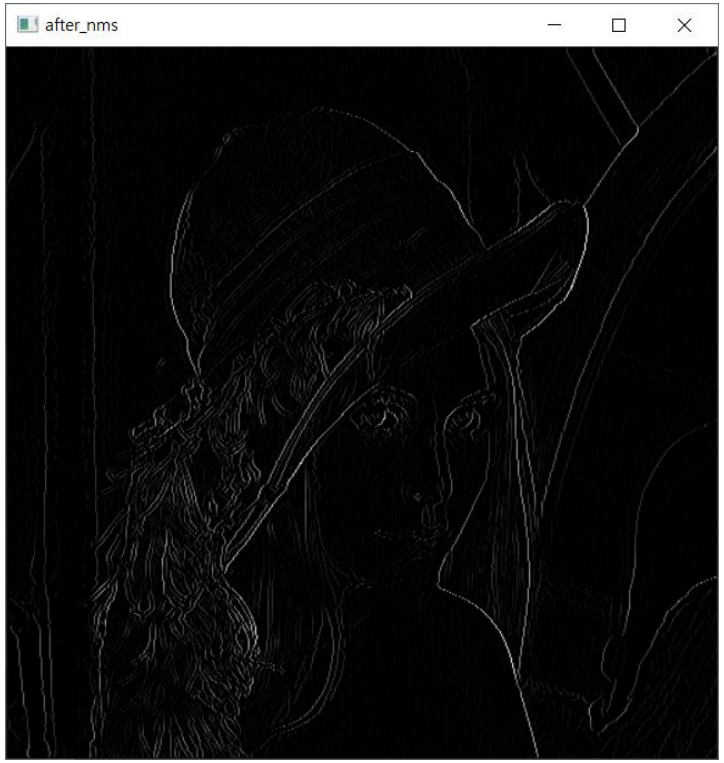
과제

- **Canny edge detection 구현하기**
  - ➢ double thresholding 수행
  - ➢ threshold는 정해진 값을 사용할 예정
    - ➢ 본인이 직접 바꾸면서 실험 권장
  - ➢ 8-neighbor 사용

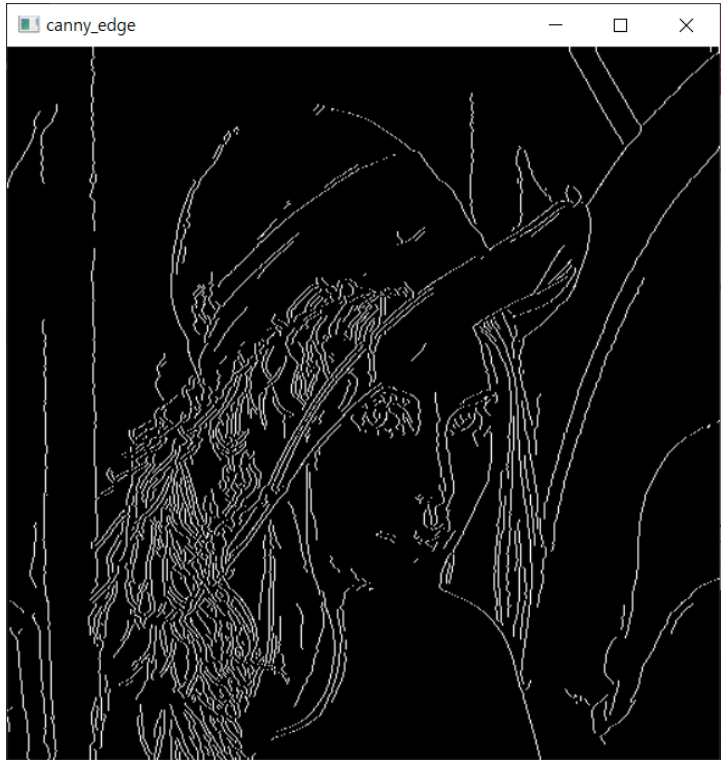
```python
def double_thresholding(src):
    dst = src.copy()

    # 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
    for row in range(h):
        for col in range(w):
            pass

    # return dst
    dst = dst.astype('float32') / 255.0
    return dst[1:-1, 1:-1]
```

과제

- **Canny edge detection 구현하기**
  - ➢ double thresholding 수행



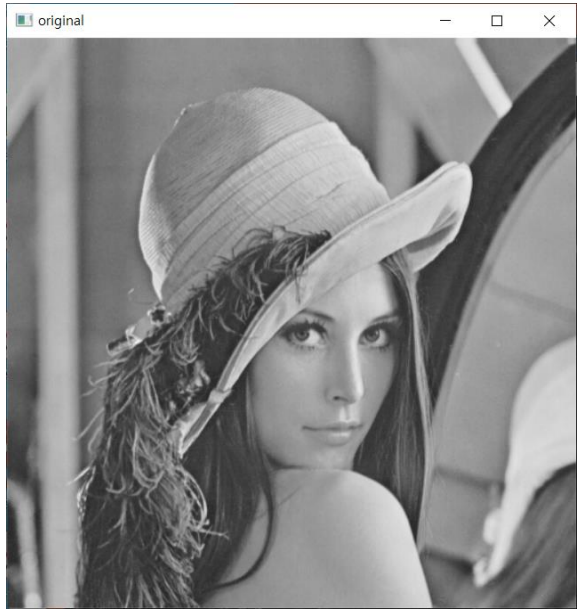
before



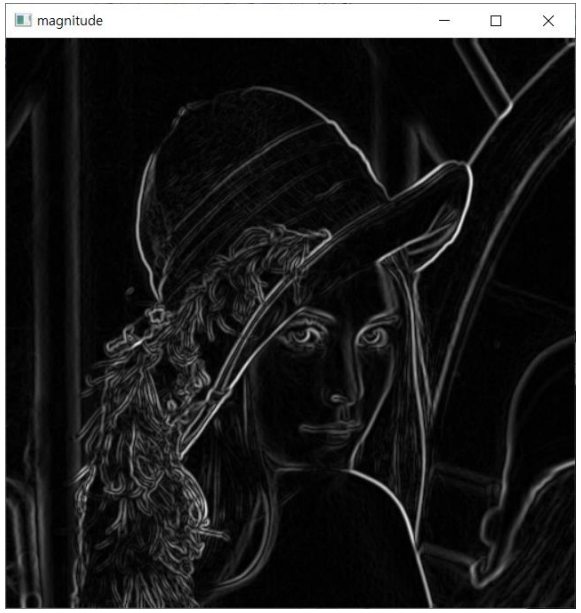
after

과제

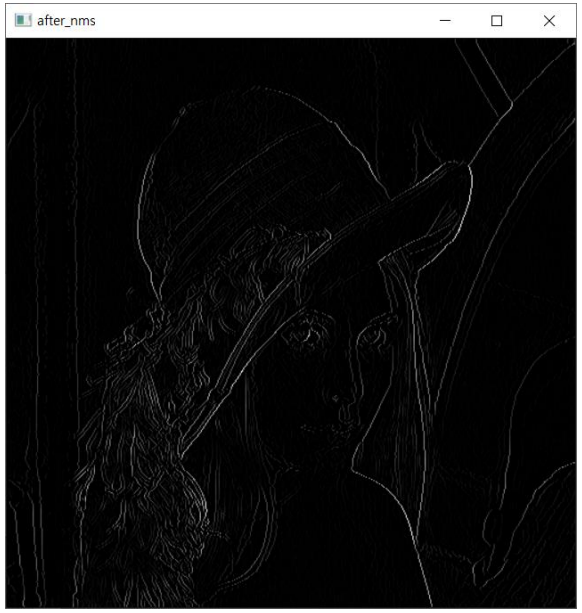
- **Canny edge detection 구현하기**
  - ➢ 결과물



original



magnitude



after nms

canny edge

과제

- **canny_edge_detection.py**
  - 6개의 함수를 구현해서 canny edge detection을 완성
    - apply_gaussian_filter
    - apply_sobel_filter
    - calc_magnitude
    - calc_angle
    - non_maximum_suppression
    - double_thresholding

  - 보고서에 결과 4개의 이미지를 포함하여 작성
    - Original, magnitude, after_nms, canny_edge
    - main에 시각화 코드가 주어져 있음

과제

- **제출 방법**
  - 코드 파일
    - 구현 결과가 포함된 python 파일(.py)
  - 보고서
    - [IP]201900000_홍길동_2주차_과제.pdf
    - 보고서 양식 사용
    - PDF 파일 형식으로 제출(pdf가 아닌 다른 양식으로 제출시 감점)
  - 제출 파일
    - [IP]201900000_홍길동_2주차_과제.zip
    - .py 파일과 pdf 보고서를 하나의 파일로 압축한 후, 양식에 맞는 이름으로 제출

출석체크

- **Zoom 퇴장 전, [학번 이름]을 채팅창에 올린 후 퇴장해 주시기 바랍니다.**

QnA