Computer Graphics
실습 5.

2020. 10. 14

박 화 종

aqkrghkwhd@naver.com

공 지 사 항

- 과제 제출 시 .pdf 혹은 .py 파일을 첨부하지 않으면 감점(-6점)
- .pdf 파일이 아니라 .hwp 혹은 .docx(워드) 로 보고서 제출 시 감점(-1점)

실습 소개

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

- **TA 연락처**
  - 박화종
  - 공대 5호관 506호 컴퓨터비전 연구실
  - aqkrghkwhd@naver.com

- **실습 튜터**
  - 최수민(00반)
  - eocjstnals12@naver.com
  - 신준호(01반)
  - wnsgh578@naver.com

목 차

3주차 과제 리뷰

- Canny edge detection 구현하기

original

Canny Edge Detection

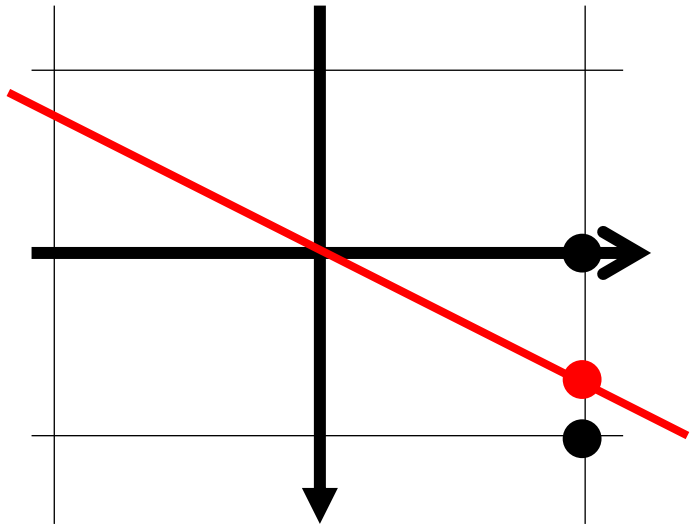
3주차 과제 리뷰

```python
# Ix와 Iy의 magnitude를 구함
def calcMagnitude(Ix, Iy):
    ###########################################
    # TODO                                    #
    # calcMagnitude 완성                      #
    # magnitude : ix와 iy의 magnitude         #
    ###########################################
    # Ix와 Iy의 magnitude를 계산
    magnitude = np.sqrt(Ix ** 2 + Iy ** 2)
    return magnitude


# Ix와 Iy의 angle을 구함
def calcAngle(Ix, Iy):
    ########################################
    # TODO                                 #
    # calcAngle 완성                       #
    # angle     : ix와 iy의 angle          #
    ########################################
    angle = np.rad2deg(np.arctan(Iy / (Ix+1e-6)))

    return angle
```

3주차 과제 리뷰

Non maximum suppression 함수

tan(Θ)



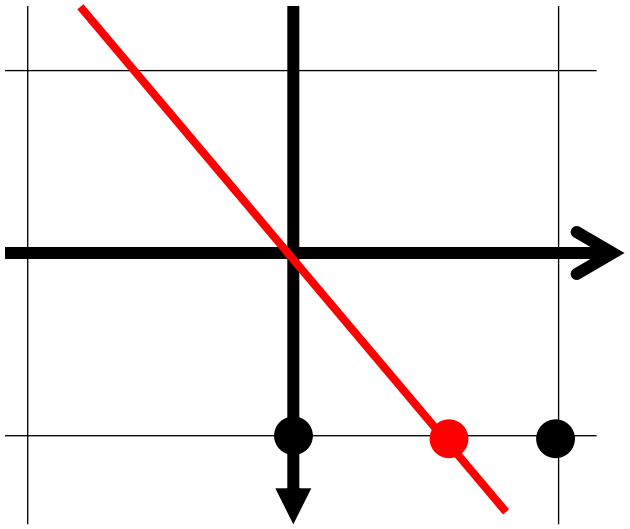
0 ~ 45

```python
# 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 = (1 - rate) * magnitude[row - 1, col] + rate * magnitude[row - 1, col + 1]
    down_magnitude = (1 - rate) * magnitude[row + 1, col] + rate * magnitude[row + 1, col - 1]
    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 = (1 - rate) * magnitude[row, col - 1] + rate * magnitude[row + 1, col - 1]
    right_magnitude = (1 - rate) * magnitude[row, col + 1] + rate * magnitude[row - 1, 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 = (1 - rate) * magnitude[row - 1, col] + rate * magnitude[row - 1, col - 1]
    down_magnitude = (1 - rate) * magnitude[row + 1, col] + rate * magnitude[row + 1, col + 1]
    if magnitude[row, col] == max(up_magnitude, magnitude[row, col], down_magnitude):
        larger_magnitude[row, col] = magnitude[row, col]
```

3주차 과제 리뷰

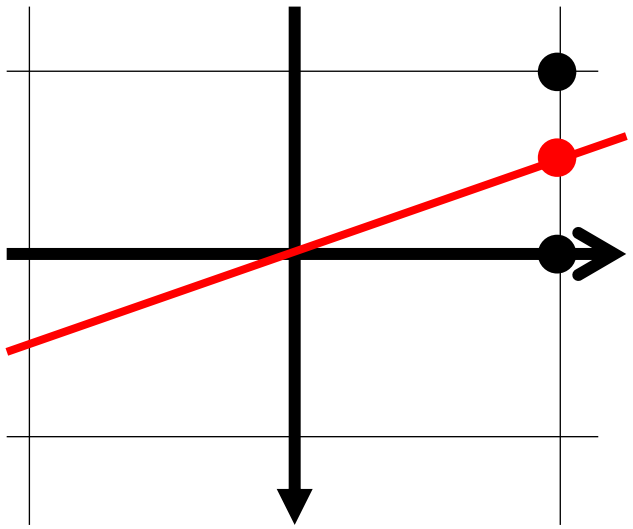
Non maximum suppression 함수

$$\frac{1}{\tan(\Theta)}$$



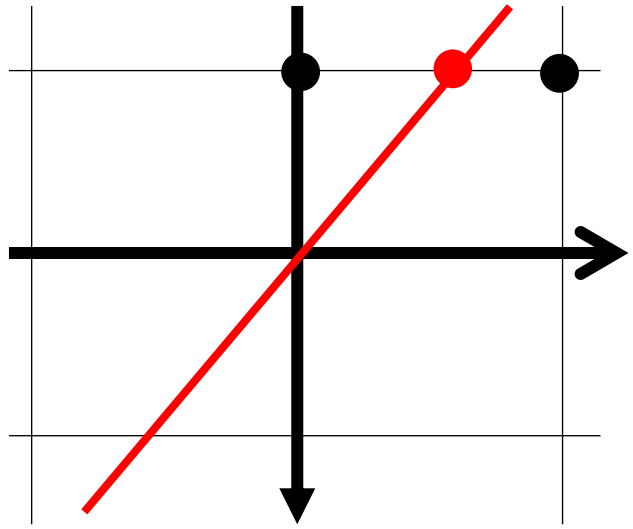
45 ~ 90

$$-\tan(\Theta)$$



0 ~ -45

$$-\frac{1}{\tan(\Theta)}$$



-45 ~ -90

3주차 과제 리뷰

Non maximum suppression 함수

```python
# double_thresholding 수행 high threshold value는 내장함수(otsu방식 이용)를 사용하여 구하고 low thr
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

    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:
                weak_edge = []
                weak_edge.append((row, col))
                search_weak_edge(dst, weak_edge, high_threshold_value, low_threshold_value)
                if calssity_edge(dst, weak_edge, high_threshold_value):
                    for idx in range(len(weak_edge)):
                        (r, c) = weak_edge[idx]
                        dst[r, c] = 255
                else:
                    for idx in range(len(weak_edge)):
                        (r, c) = weak_edge[idx]
                        dst[r, c] = 0

    return dst
```

3주차 과제 리뷰

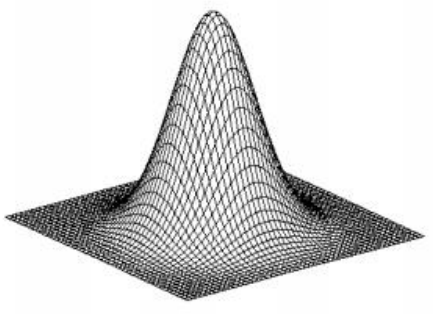
Non maximum suppression 함수

```
def search_weak_edge(dst, edges, high_threshold_value, low_threshold_value):
    # now (row, col) = edges[-1]
    (row, col) = edges[-1]
    for i in range(-1, 2):
        for j in range(-1, 2):
            if dst[row+i, col+j] < high_threshold_value and dst[row+i, col+j] >= low_threshold_value:
                #중복 아닌 값만 append
                if edges.count((row+i, col+j)) < 1:
                    edges.append((row+i, col+j))
                    search_weak_edge(dst, edges, high_threshold_value, low_threshold_value)


def calssity_edge(dst, weak_edge, high_threshold_value):
    for idx in range(len(weak_edge)):
        (row, col) = weak_edge[idx]
        value = np.max(dst[row-1:row+2, col-1:col+2])
        if value >= high_threshold_value:
            return True
```

실습

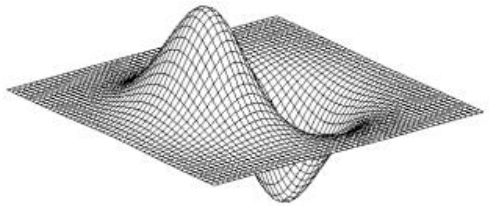
- **Filter 모양 확인**
  - Average filter
  - Gaussian filter
  - Derivative of Gaussian

2D Derivative of Gaussian (DoG)



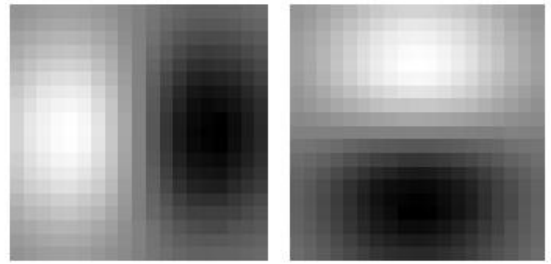
Gaussian

$$G(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$



derivative of Gaussian (DOG)

$$\nabla G(x,y)=\left(G_x,G_y\right)$$



실습

- Average filter

```
[[0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]]
```

```python
import cv2
import numpy as np

def show_filter(type, fshape):
    show_filter_size = (32, 32)
    if type == 'average':
        filter = np.ones(fshape)
        filter = filter / (fshape[0] * fshape[1])
        print(filter)

    cv2.imshow(type + ' filter', filter)
    cv2.waitKey()
    cv2.destroyAllWindows()

def main():
    type = 'average'
    fshape = (3, 3)
    show_filter(type, fshape)

if __name__ =='__main__':
    main()
```

실습

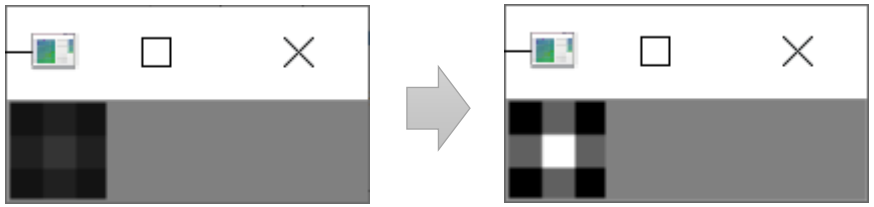
- Average filter

```
[[0.11111111 0.11111111 0.11111111 ... 0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111 ... 0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111 ... 0.11111111 0.11111111 0.11111111]
 ...
 [0.11111111 0.11111111 0.11111111 ... 0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111 ... 0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111 ... 0.11111111 0.11111111 0.11111111]]
```

```python
import cv2
import numpy as np

def show_filter(type, fshape):
    show_filter_size = (32, 32)
    if type == 'average':
        filter = np.ones(fshape)
        filter = filter / (fshape[0] * fshape[1])
        filter = cv2.resize(filter, (show_filter_size), interpolation=cv2.INTER_NEAREST)

        print(filter)

    cv2.imshow(type + ' filter', filter)
    cv2.waitKey()
    cv2.destroyAllWindows()

def main():
    type = 'average'
    fshape = (3, 3)
    show_filter(type, fshape)

if __name__ =='__main__':
    main()
```

실습

- Gaussian



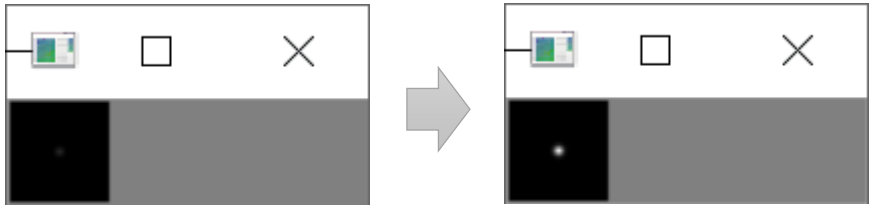
normalize(보기편하게)

```
<gaussian 3x3>
[[0.07511361 0.1238414  0.07511361]
 [0.1238414  0.20417996 0.1238414 ]
 [0.07511361 0.1238414  0.07511361]]
<gaussian 32x32 Nearest>
[[0.07511361 0.07511361 0.07511361 ... 0.07511361 0.07511361 0.07511361]
 [0.07511361 0.07511361 0.07511361 ... 0.07511361 0.07511361 0.07511361]
 [0.07511361 0.07511361 0.07511361 ... 0.07511361 0.07511361 0.07511361]
 ...
 [0.07511361 0.07511361 0.07511361 ... 0.07511361 0.07511361 0.07511361]
 [0.07511361 0.07511361 0.07511361 ... 0.07511361 0.07511361 0.07511361]
 [0.07511361 0.07511361 0.07511361 ... 0.07511361 0.07511361 0.07511361]]
<gaussian 32x32 Normalize 0 ~ 1>
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

```python
import cv2
import numpy as np

def my_get_Gaussian_filter(fshape, sigma=1):
    (f_h, f_w) = fshape
    y, x = np.mgrid[-(f_h // 2):(f_h // 2) + 1, -(f_w // 2):(f_w // 2) + 1]
    #2차 gaussian mask 생성
    filter_gaus =  1 / (2 * np.pi * sigma**2) * np.exp(-(( x**2 + y**2 )/(2 * sigma**2)))
    #mask의 총 합 = 1
    filter_gaus /= np.sum(filter_gaus)
    return filter_gaus

def show_filter(type, fshape):
    show_filter_size = (32, 32)
    if type == 'average':...

    elif type == 'gaussian':
        sigma = 1
        filter = my_get_Gaussian_filter(fshape, sigma)
        print('<gaussian 3x3>')
        print(filter)
        filter = cv2.resize(filter, (show_filter_size), interpolation=cv2.INTER_NEAREST)
        print('<gaussian 32x32 Nearest>')
        print(filter)
        filter = (filter - np.min(filter)) / (np.max(filter) - np.min(filter))
        print('<gaussian 32x32 Normalize 0 ~ 1>')
        print(filter)

    cv2.imshow(type + ' filter', filter)
    cv2.waitKey()
    cv2.destroyAllWindows()

def main():
    type = 'gaussian'
    fshape = (3, 3)
    show_filter(type, fshape)

if __name__ =='__main__':
    main()
```

실습

- Gaussian



normalize(보기편하게)

```
<gaussian 32x32>
[[1.05301064e-112 5.67540985e-106 1.12529714e-099 ... 1.12529714e-099
  5.67540985e-106 1.05301064e-112]
 [5.67540985e-106 3.05887478e-099 6.06501231e-093 ... 6.06501231e-093
  3.05887478e-099 5.67540985e-106]
 [1.12529714e-099 6.06501231e-093 1.20254593e-086 ... 1.20254593e-086
  6.06501231e-093 1.12529714e-099]
 ...
 [1.12529714e-099 6.06501231e-093 1.20254593e-086 ... 1.20254593e-086
  6.06501231e-093 1.12529714e-099]
 [5.67540985e-106 3.05887478e-099 6.06501231e-093 ... 6.06501231e-093
  3.05887478e-099 5.67540985e-106]
 [1.05301064e-112 5.67540985e-106 1.12529714e-099 ... 1.12529714e-099
  5.67540985e-106 1.05301064e-112]]
```

```python
def show_filter(type, fshape):
    show_filter_size = (32, 32)
    if type == 'average':...

    elif type == 'gaussian':
        sigma = 1
        """filter = my_get_Gaussian_filter(fshape, sigma)
        print('<gaussian 3x3>')
        print(filter)
        filter = cv2.resize(filter, (show_filter_size), interpolation=cv2.INTER
        print('<gaussian 32x32 Nearest>')
        print(filter)"""

        filter = my_get_Gaussian_filter(fshape, sigma)
        print('<gaussian 32x32>')
        print(filter)

        filter = (filter - np.min(filter)) / (np.max(filter) - np.min(filter))
        print('<gaussian 32x32 Normalize 0 ~ 1>')
        print(filter)

    cv2.imshow(type + ' filter', filter)
    cv2.waitKey()
    cv2.destroyAllWindows()

def main():
    type = 'gaussian'
    fshape = (32, 32)
    show_filter(type, fshape)

if __name__ =='__main__':
    main()
```

실습

- Gaussian

normalize(보기편하게)

sigma : 3

sigma : 1

```python
def show_filter(type, fshape):
    show_filter_size = (32, 32)
    if type == 'average':...

    elif type == 'gaussian':
        sigma = 3
        """filter = my_get_Gaussian_filter(fshape, sigma)
        print('<gaussian 3x3>')
        print(filter)
        filter = cv2.resize(filter, (show_filter_size), interpolation=cv2.INTER_NEAREST)
        print('<gaussian 32x32 Nearest>')
        print(filter)"""

        filter = my_get_Gaussian_filter(fshape, sigma)
        print('<gaussian 32x32>')
        print(filter)

        filter = (filter - np.min(filter)) / (np.max(filter) - np.min(filter))
        print('<gaussian 32x32 Normalize 0 ~ 1>')
        print(filter)

    cv2.imshow(type + ' filter', filter)
    cv2.waitKey()
    cv2.destroyAllWindows()
```

실습

- Gaussian

sigma = 3

sigma = 7

sigma = 10

실습

- Gaussian filter의 Sigma값이 크면 Average필터와 비슷하다?

```
[[0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]]
```

Average filter 3x3

```
[[0.07511361 0.1238414  0.07511361]
 [0.1238414  0.20417996 0.1238414 ]
 [0.07511361 0.1238414  0.07511361]]
```

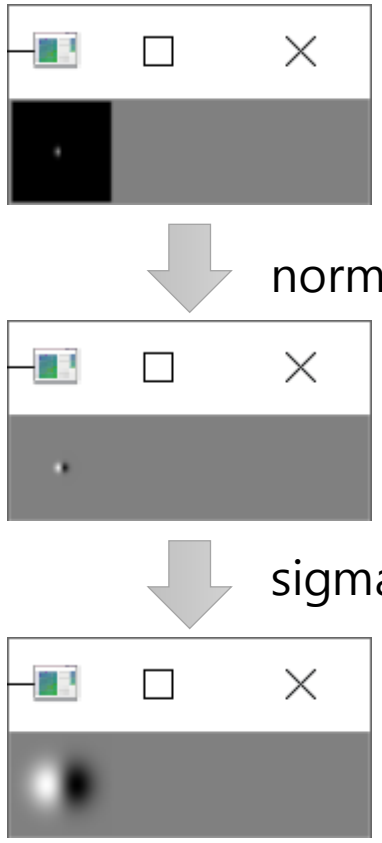
Gaussian filter 3x3
sigma : 1

```
[[0.11110741 0.11111296 0.11110741]
 [0.11111296 0.11111852 0.11111296]
 [0.11110741 0.11111296 0.11110741]]
```

Gaussian filter 3x3
sigma : 100

실습

- Derivative of Gaussian



normalize(보기편하게)

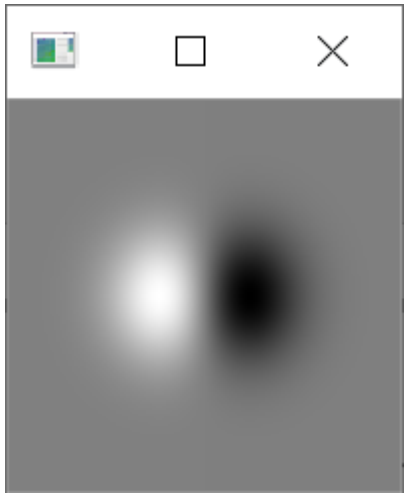
sigma : 1-> 5

```
def get_my_DoG(fshape, sigma=1):
    (f_h, f_w) = fshape
    y, x = np.mgrid[-(f_h // 2):(f_h // 2) + 1, -(f_w // 2):(f_w // 2) + 1]
    DoG_x = (-x / sigma**2) * np.exp(-(x **2 + y **2)/(2 * sigma**2))
    DoG_y = (-y / sigma**2) * np.exp(-(x **2 + y **2)/(2 * sigma**2))
    return DoG_x, DoG_y

def show_filter(type, fshape):
    show_filter_size = (32, 32)
    if type == 'average':...
    elif type == 'gaussian':...
    elif type == 'dog':
        sigma = 1
        DoG_x, DoG_y = get_my_DoG(fshape, sigma)
        filter = DoG_x
        print('<DoG 32x32>')
        print(filter)

    #filter = (filter - np.min(filter)) / (np.max(filter) - np.min(filter
    #print('<DoG 32x32 Normalize 0 ~ 1>')
    #print(filter)

    cv2.imshow(type + ' filter', filter)
    cv2.waitKey()
    cv2.destroyAllWindows()

def main():
    type = 'dog'
    fshape = (32, 32)
    show_filter(type, fshape)
```

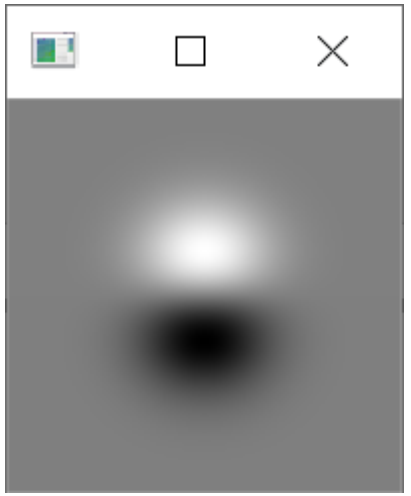
normalize

실습

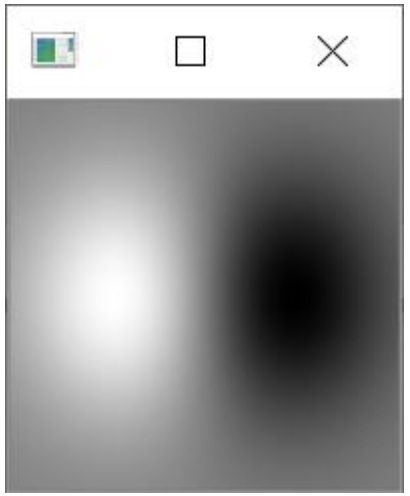
- Derivative of Gaussian



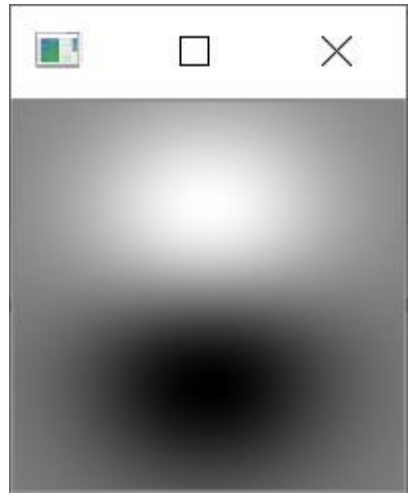
DoG_x
fshape : (128, 128)
sigma : 15



DoG_y
fshape : (128, 128)
sigma : 15



DoG_x
fshape : (128, 128)
sigma : 30



DoG_y
fshape : (128, 128)
sigma : 30

과제

```
start!
src.shape :  (552, 435, 3)
fsize :  5
M_harris time :  6.1276217
make integral image time :  0.8241929999999993
M_harris integral time :  1.1511037999999996
```

- **Integral image 사용시 속도 차이 확인**

자신의 학번을 추가하기

original

harris corner

harris corner
integral image

과제

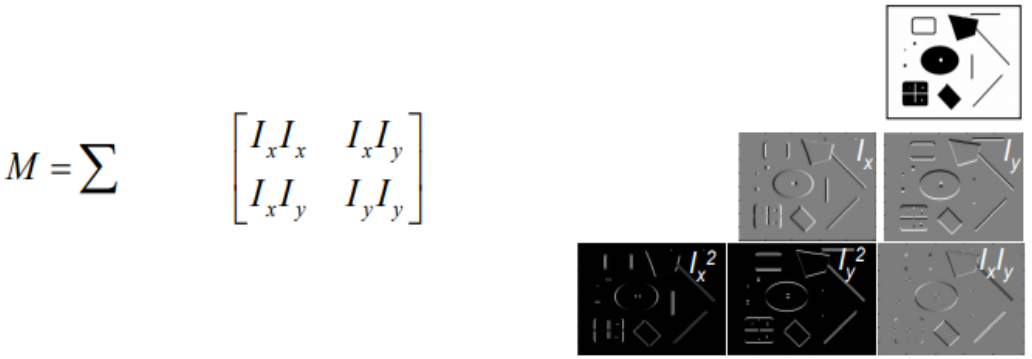
- **Integral image 사용시 속도 차이 확인**

시간이 오래 걸림

fsize : 5기준
552*435*24*3
= 17,288,640

552 : h
435 : w
24 : fsize*fsize-1
3 : Ixlx, Ixly, Iyly

$$M = \sum \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

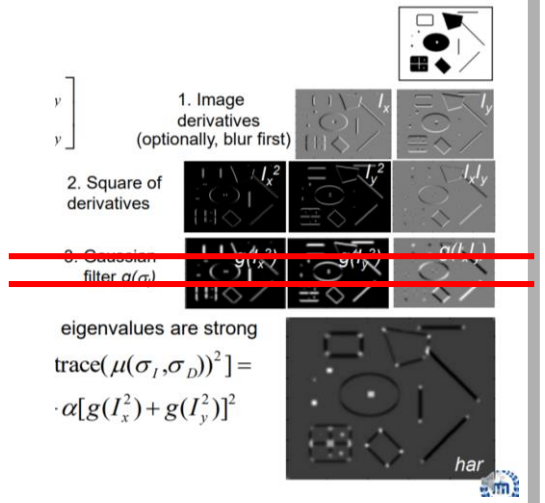


Integral image를 사용하여 시간 단축

fsize : 5기준
552*435*3*3 = 2,161,080

552 : h
435 : w
3 : D – B - C + A
3 : Ixlx, Ixly, Iyly

과제



- **Harris Corner Detection 구현하기**
  - 저번주차 과제를 참고하여 IxIx, IxIy, IyIy 구하는 부분 가지고 오기
  - G_IxIx, G_IxIy, G_IyIy는 사용하지 않음
  - integral image를 사용하지 않고 M 구하기(4중for문 사용해야함)
  - integral image를 사용하고 M 구하기

  - 각각 구한 M을 가지고 harris corner 완성시키기(저번주차 과제 참고)
  - R을 구할 때는 교수님 이론ppt 65~66page 참고(4주차 이론ppt)

Harris & Stephens (1988)

$$R = \det(M) - \kappa \mathrm{trace}^2(M)$$

Nobel (1998)

$$R = \frac{\det(M)}{\mathrm{trace}(M) + \epsilon}$$

$$det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$$

k : 0.04
E : 1E-8

$$trace\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = a + d$$

과제 - 구현

- 제공된 코드가 제대로 동작하도록 구현하기
  - integral image를 사용하지 않고 M 구한 후 harris corner 완성
  - integral image를 사용하고 M 구한 후 harris corner 완성

- 사용한 hyperparameter
  - sobel filter size = 3x3
  - M 구할 때의 filter size = 5x5
  - k = 0.04
  - E = 1E-8
  - threshold rate = 0.01
  - local maxima filter size = 21

  - R = Harris & Stephens (1988) (Nobel (1998)로 구현해도 상관없음)

과제 - 보고서

- **보고서**
  - **내용:**
    - 이름, 학번, 학과
    - 구현 코드: 구현한 코드
    - 코드 설명: 구현한 코드에 대한 설명(설명은 1page를 넘기지 말 것, 1줄이어도 상관없음)
    - 이미지: 과제 첫 페이지를 참고하여 이미지 4개 첨부(시간측정결과 포함)
    - 느낀 점 : 결과를 보고 느낀 점, 혹은 과제를 하면서 어려웠던 점 등
    - 과제 난이도: 개인적으로 생각하는 난이도 (과제가 너무 쉬운 것 같다 등)

  - .pdf 파일로 제출 (이 외의 파일 형식일 경우 감점)

  - **파일 이름:**
    - [CG]20xxxxxxx_이름_n주차_과제.pdf

과제

- **제출 기한**
  - 10월 27일 23시 59분까지 (최대 점수 10점)
- **추가 제출 기한**
  - 11월 03일 23시 59분까지 (최대 점수 4점, 과제 총점 계산 후 -6점)
  - 11월 04일 00시 00분 이후 (점수 0점)
- **채점**
  - 구현을 못하거나(잘못 구현하거나) 보고서 내용이 빠진 경우 감점
  - 아무것도 구현하지 못해도 과제 제출하면 기본점수 있음
  - 다른 사람의 코드를 copy해서 제출시 보여준 사람, copy한 사람 둘 다 0점
  - 내장함수 사용시 감점(내장함수를 사용해도 된다고 말 한 것 제외) – cv2.cornerHarris 사용 불가
  - 저번주차 자신이 구현한 코드 가지고 오는 건 상관없음
- **제출 파일**
  - 아래의 파일을 압축해서 [CG]20xxxxxxx_이름_n주차_과제.zip로 제출
    - .py 파일 전부
    - .pdf 보고서 파일

QnA