Image Processing
실습 6주차

안 준 혁

Department of Computer Science and Engineering

Chungnam National University, Korea

실습 소개

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

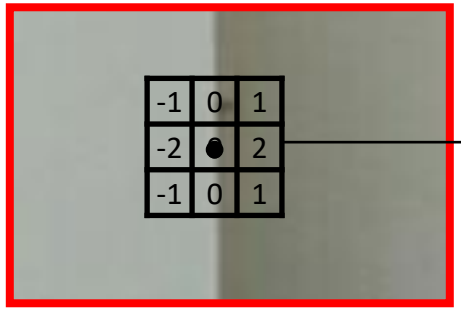
- **TA 연락처**
  - 공대 5호관 531호 컴퓨터비전 연구실
  - 과제 질문은 [IP]를 제목에 붙여 메일로 주세요.
  - 00반
    - 안준혁
    - ajh99345@gmail.com
  - 01반
    - 신동헌
    - doghon85@naver.com

목차

- **실습**
  - Sobel filter
  - 식으로 Derivative of Gaussian (DoG) 필터 만들어보기

- **과제**
  - Filtering으로 Derivative of Gaussian (DoG) 필터 만들어보기
  - Image resizing
    - Bilinear Interpolation - Pixel(1x1 area)

Sobel filter



-1	0	1
-2	0	2
-1	0	1

Sobel filter x vertical

=

1
2
1

Blurring

*

-1	0	1

1D derivative filter (x-direction)

-1	-2	-1
0	0	0
1	2	1

Sobel filter y horizontal

=

-1
0
1

1D derivative filter (y-direction)

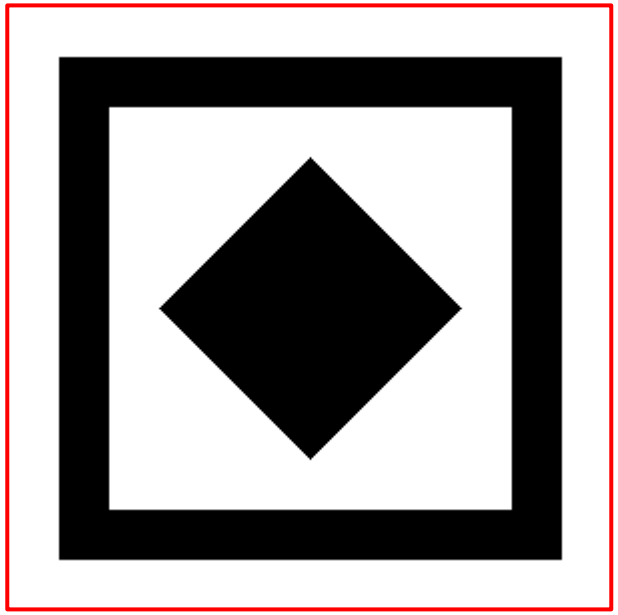
*

1	2	1

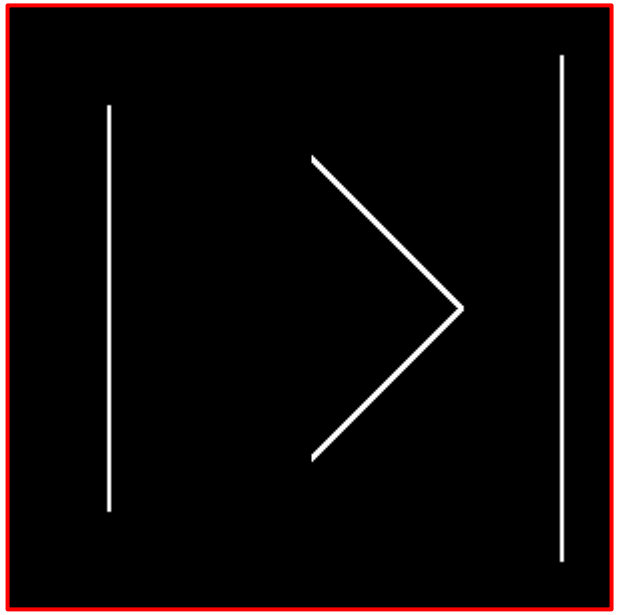
Blurring

Sobel filter

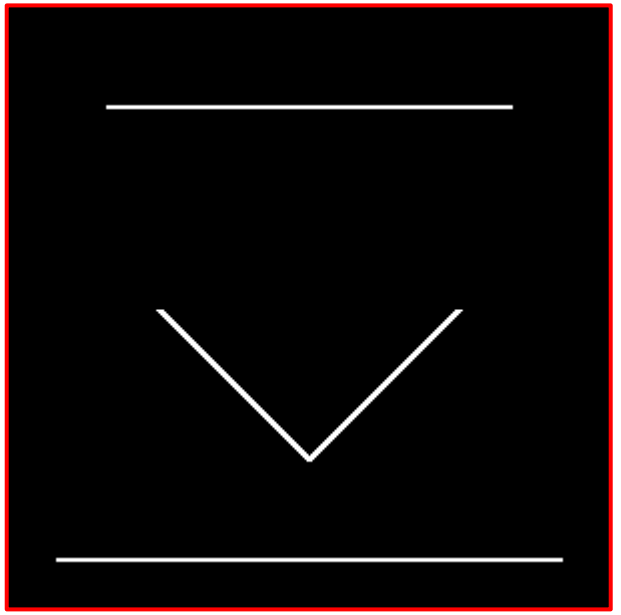
- **Example**
  - Python 시각화 코드를 통한 display



f



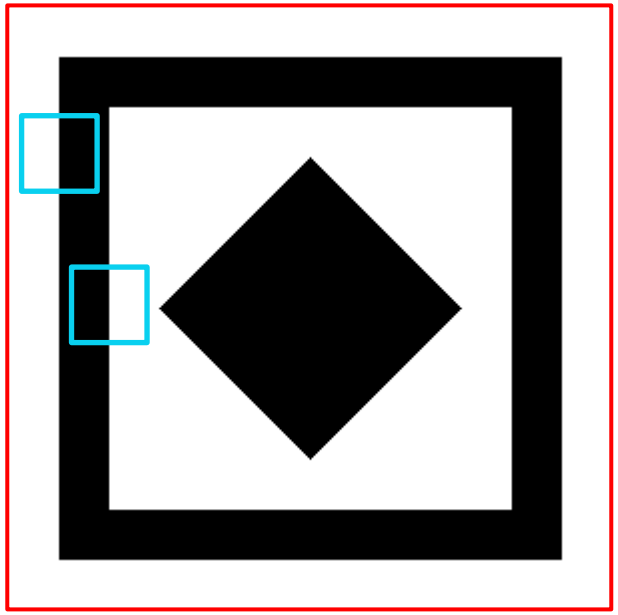
f_x



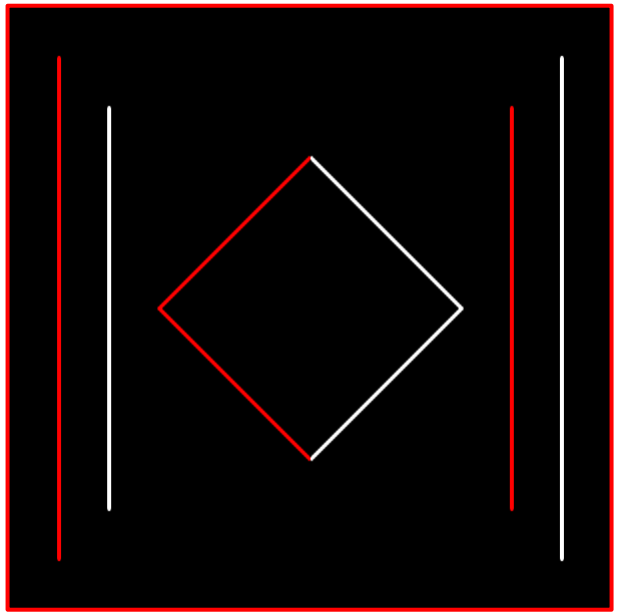
f_y

Sobel filter

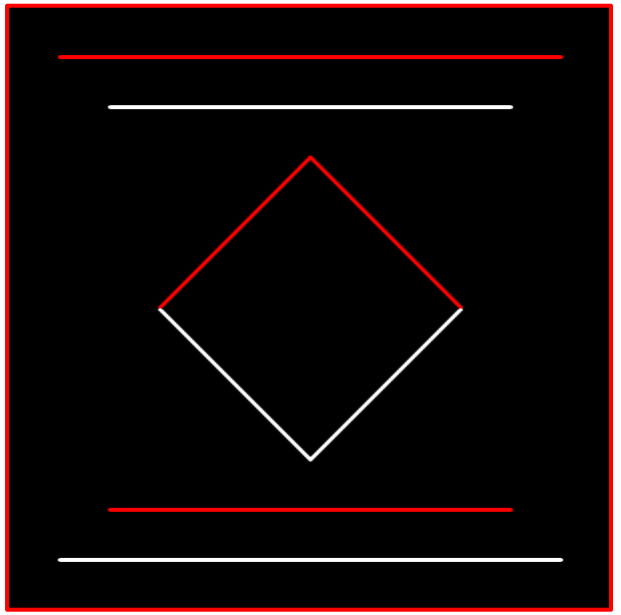
- **Example**
  - Python 시각화 코드를 통한 display



f



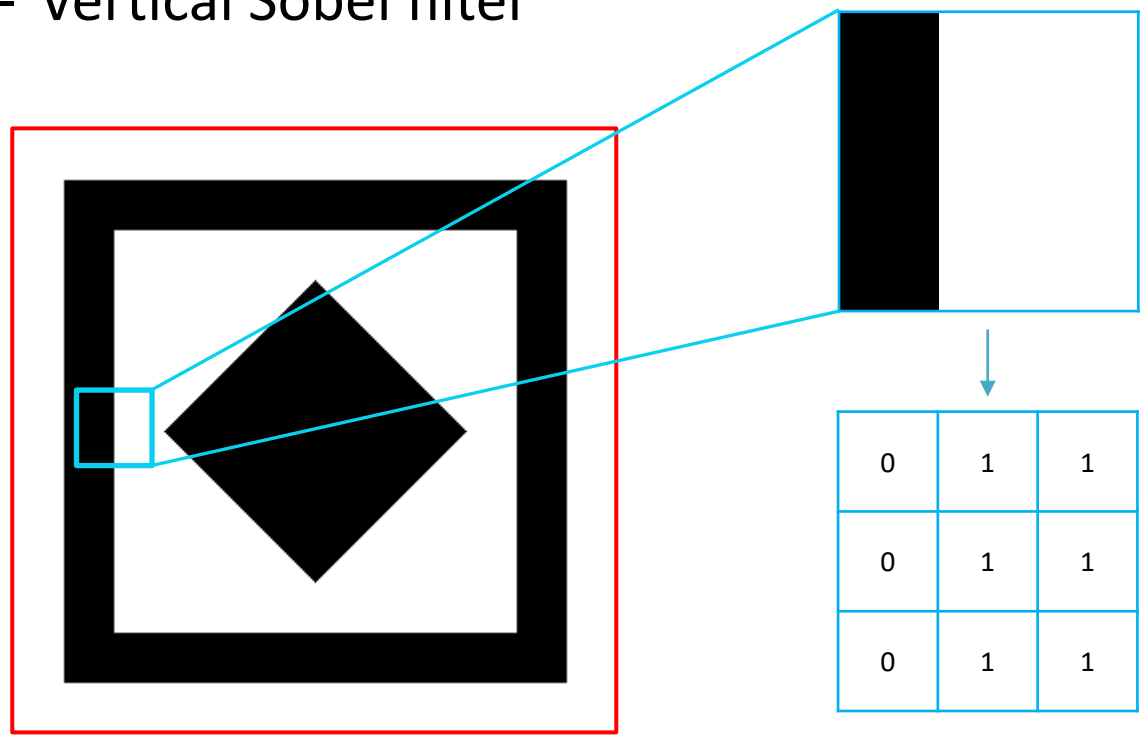
f_x



f_y

Sobel filter

- **Example**
  - Vertical Sobel filter



f

0	1	1
0	1	1
0	1	1

Input

-1	0	1
-2	0	2
-1	0	1

Kernel

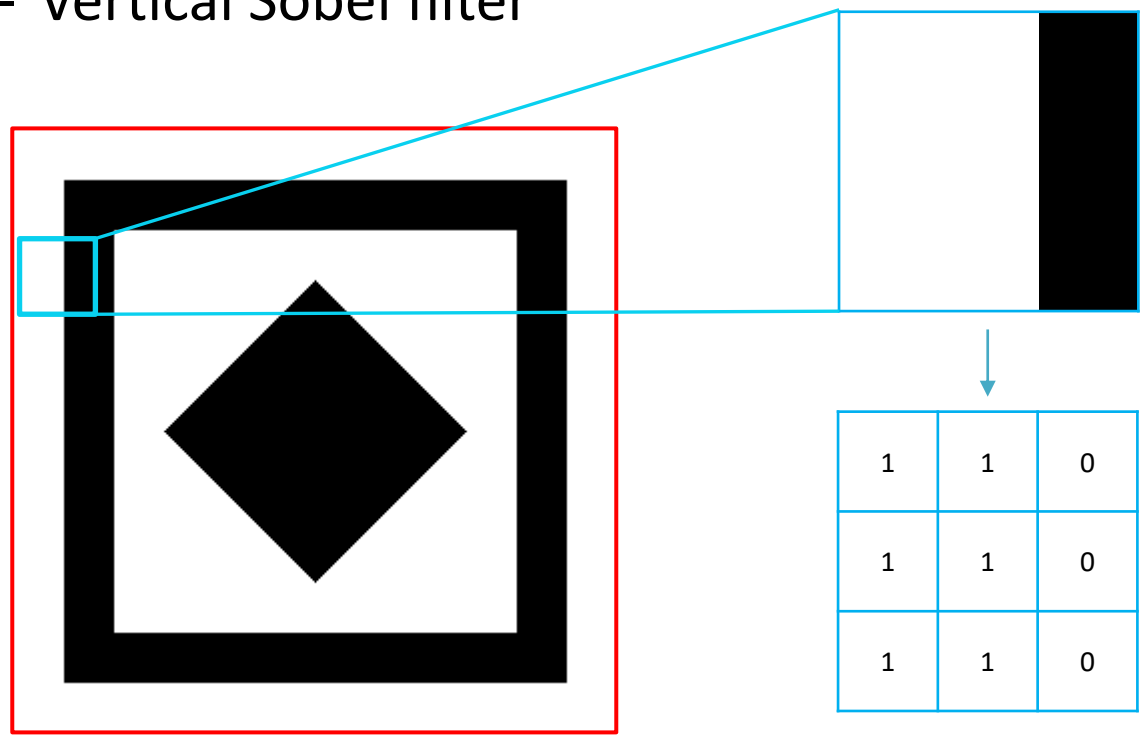
0	0	1
0	0	2
0	0	1

Output

$1 + 2 + 1 = 4$

Sobel filter

- **Example**
  - Vertical Sobel filter



f

1	1	0
1	1	0
1	1	0

Input

-1	0	1
-2	0	2
-1	0	1

Kernel

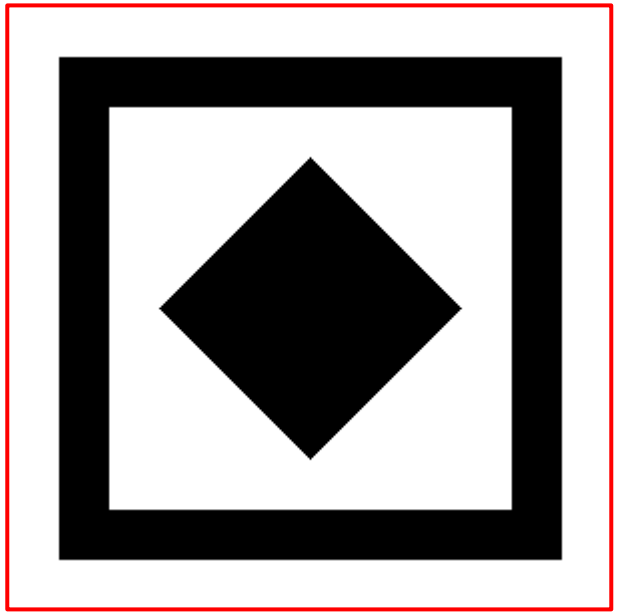
-1	0	0
-2	0	0
-1	0	0

Output

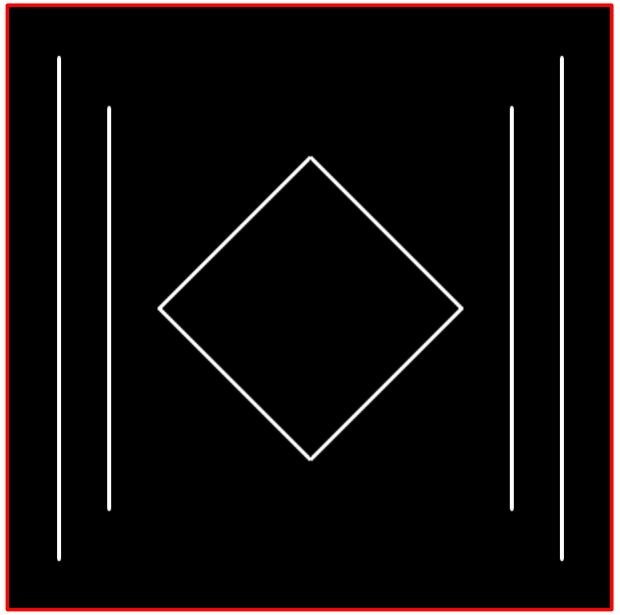
$$(-1) + (-2) + (-1) = -4$$

Sobel filter

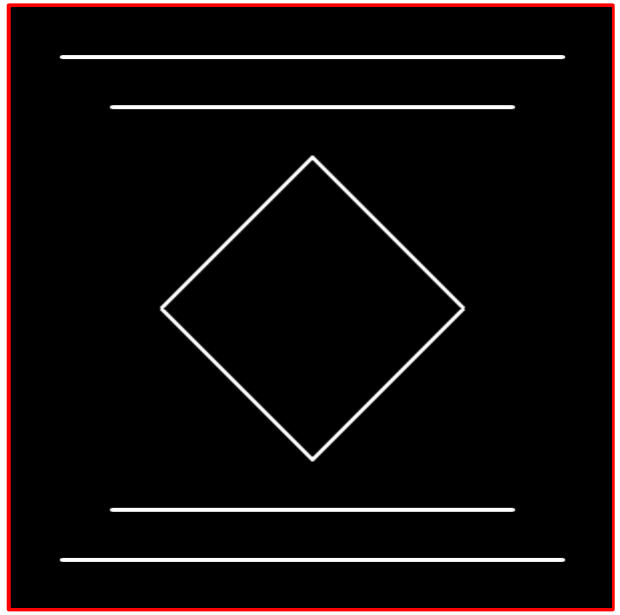
- **Example**
  - Python 시각화 코드를 통한 display



f



$|f_x|$



$|f_y|$

Sobel filter

- **Magnitude** f_x: Vertical Sobel filtering된 이미지

  – Image gradient: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

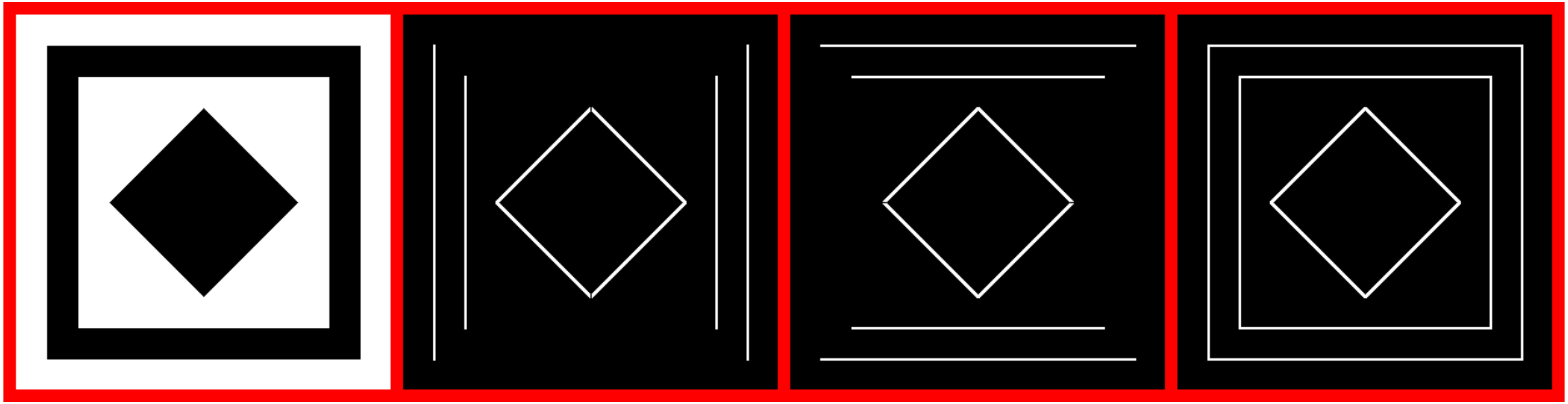
  f_y: Horizontal Sobel filtering된 이미지

  – Magnitude (edge strength): $\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

$$M = \sqrt{{f_x}^2 + {f_y}^2}$$

Sobel filter

- **Example**
  - Python 시각화 코드를 통한 display



f $|f_x|$ $|f_y|$ M

실습1 Sobel_practice.py

f
(image)

*

-1	0	1
-2	0	2
-1	0	1

$Sobel\ filter\ x$

f_x
(gradient_x)

*

-1	-2	-1
0	0	0
1	2	1

$Sobel\ filter\ y$

f_y
(gradient_y)

$$\sqrt{{f_x}^2 + {f_y}^2}$$

M
(magnitude)

Overflow 막기 위해
np.float32로 이미지 불러오기

np.abs(src):
src의 값들을 절대값 씌워줌

실습1 Sobel_practice.py

- **Lena.png**

Original

x-direction derivative

Gradient magnitude

y-direction derivative

실습1 Sobel_practice.py

- **noise_Lena.png**
  - Noise가 낀 이미지에 대해 Sobel filter 적용해보기

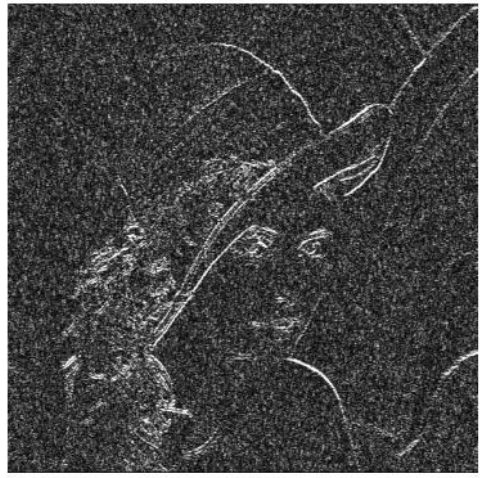
실습1 Sobel_practice.py

- **noise_Lena.png**

Original

x-direction derivative

Gradient magnitude



y-direction derivative

Derivative of Gaussian (DoG)

- **noise_Lena.png**

Original

Sobel

DoG (Equation)

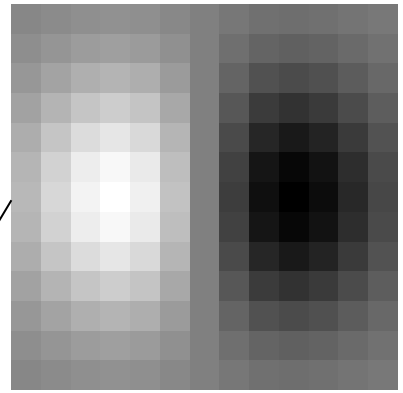
실습2 DoG_practice.py

- 식으로 DoG filter 만들어보기
  - $G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$
  - $\nabla G(x,y) = \left[\frac{\partial G}{\partial x}, \frac{\partial G}{\partial y}\right]$
  - $\frac{\partial G}{\partial x} = ?$
  - $\frac{\partial G}{\partial y} = ?$

실습2 DoG_practice.py

f
(image)

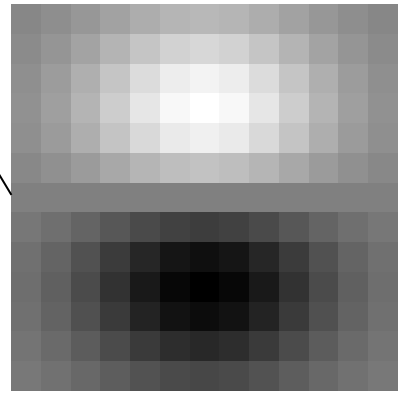
*



$DoG\ filter\ x$

f_x

*



$DoG\ filter\ y$

f_y

$$\sqrt{{f_x}^2 + {f_y}^2}$$

M
(DoG_equation)

Overflow 막기 위해
np.float32로 이미지 불러오기

실습2 DoG_practice.py

- **noise_Lena.png**

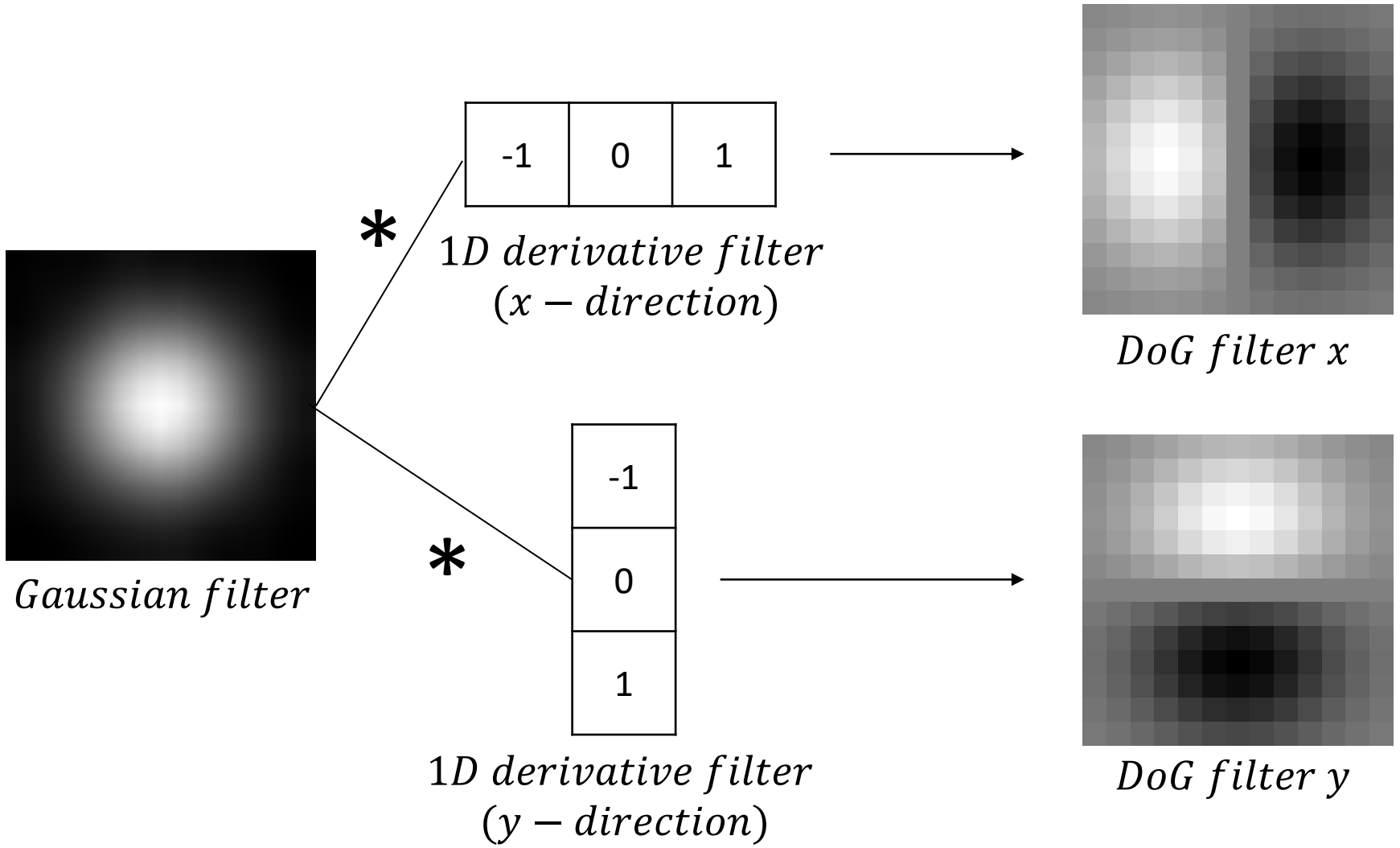
Original

Sobel

DoG (Equation)

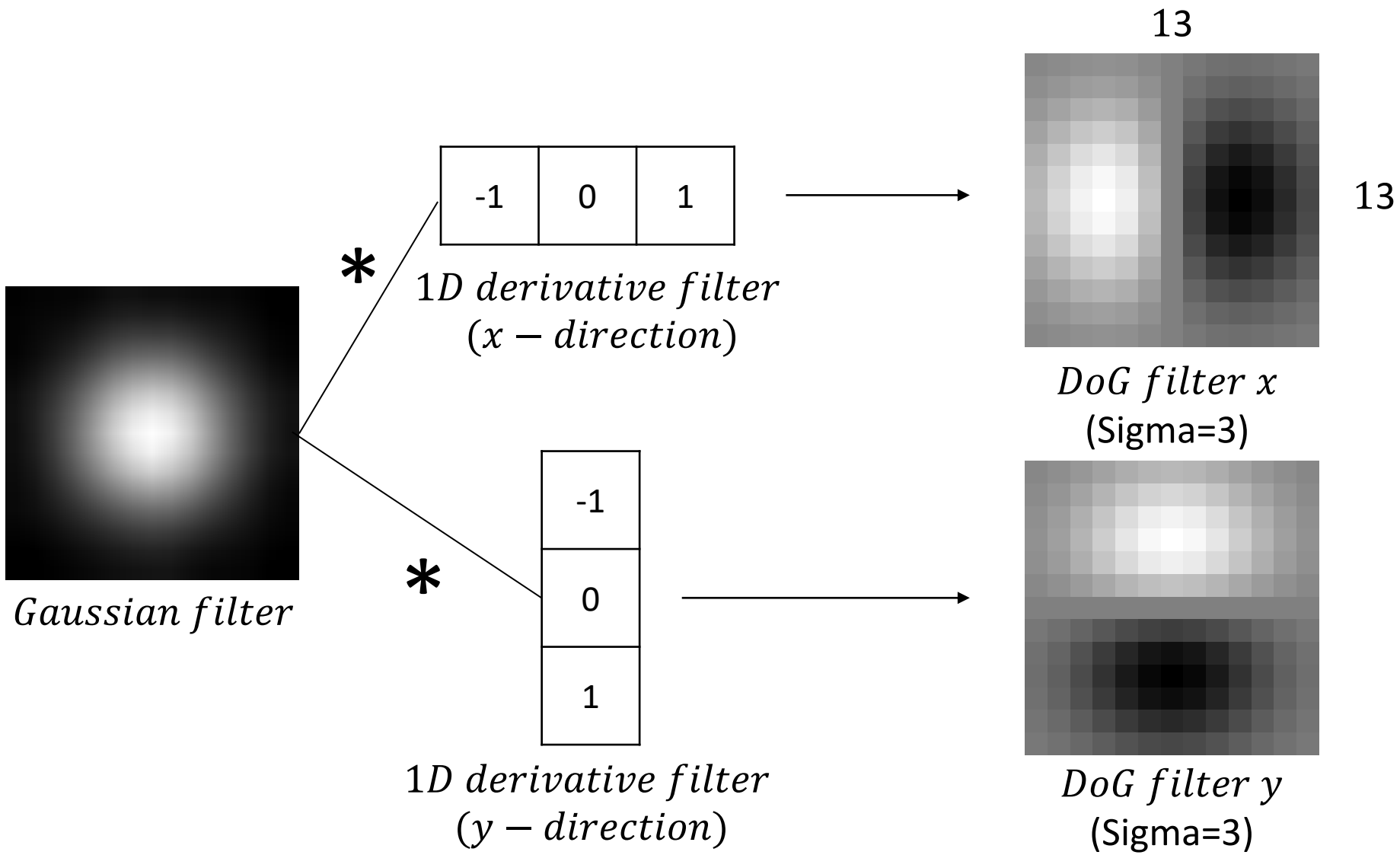
과제 DoG.py

- **Filtering으로 DoG filter 만들어보기**



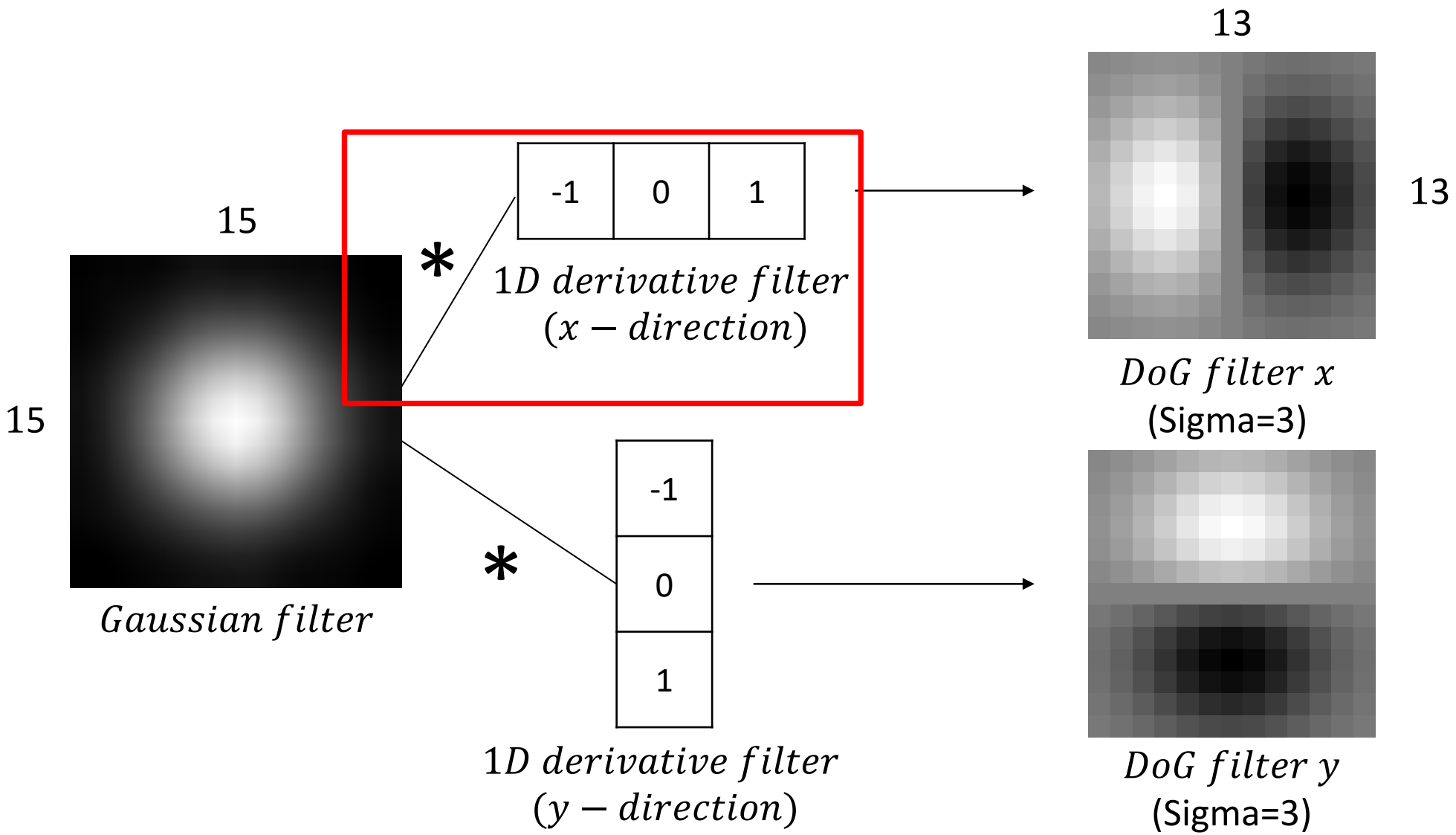
과제 DoG.py

- **Sigma값이 3인 13x13 DoG 필터 만들기**



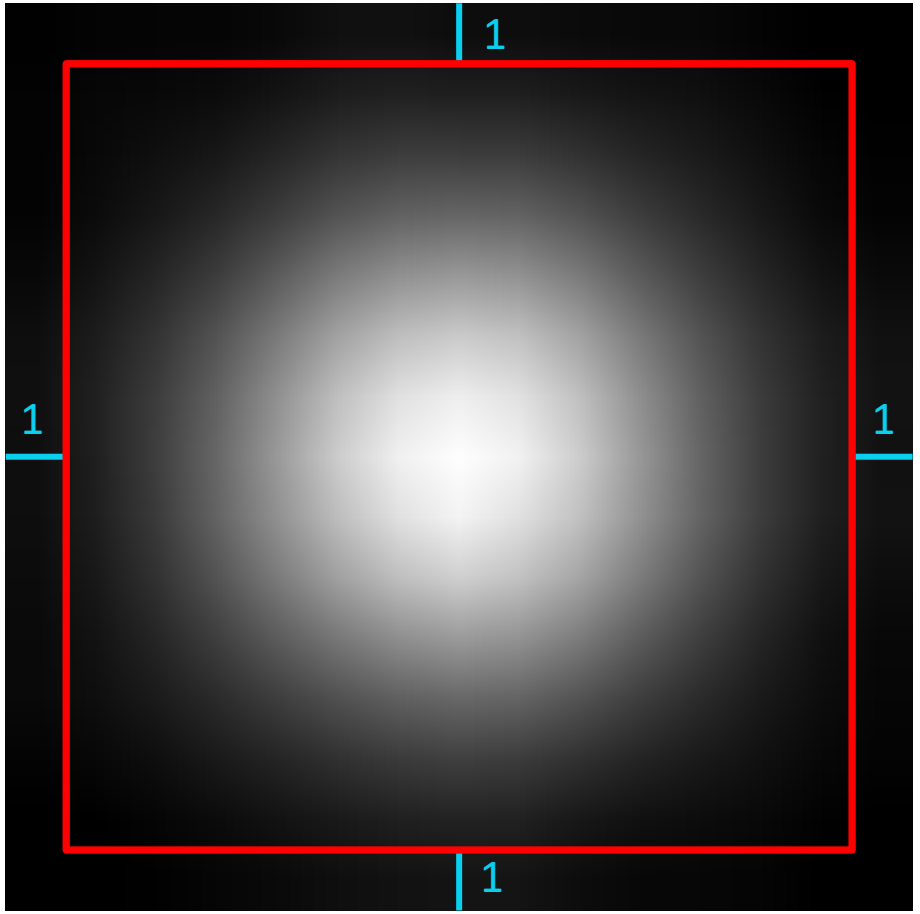
과제 DoG.py

- **Sigma값이 3인 13x13 DoG 필터 만들기**



과제 DoG.py

- **Sigma값이 3인 13x13 DoG 필터 만들기**



$Gaussian\ filter$

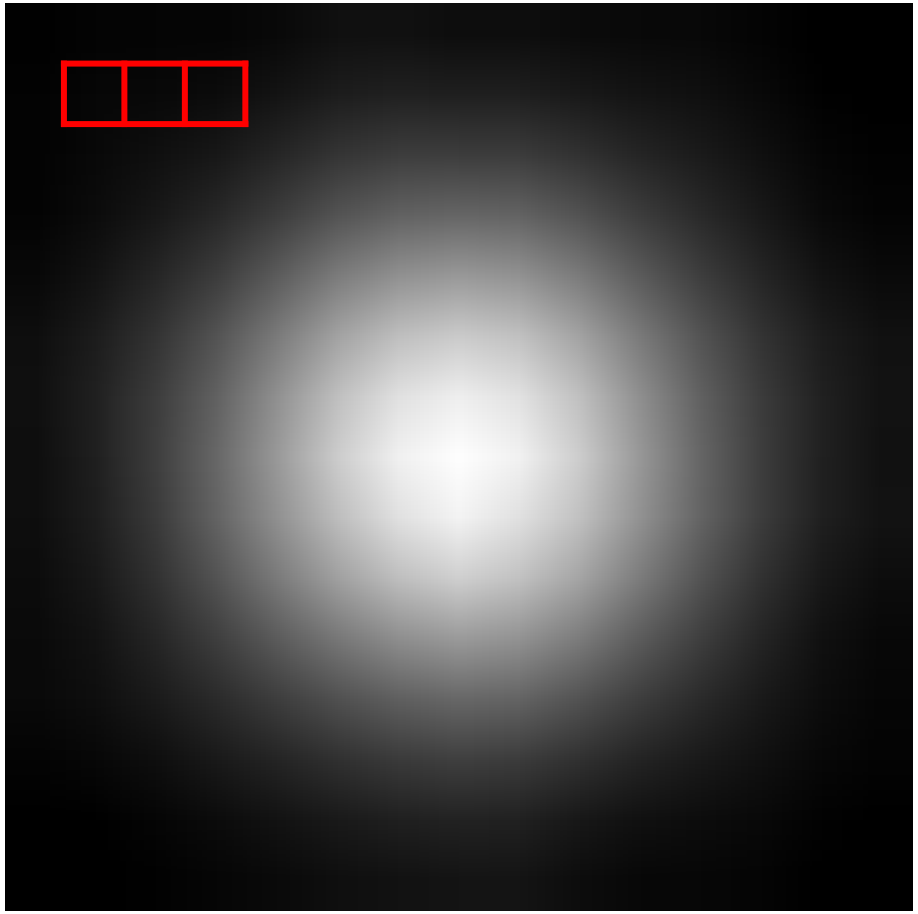
과제 DoG.py

- Sigma값이 3인 13x13 DoG 필터 만들기

$Gaussian\ filter$

과제 DoG.py

- Sigma값이 3인 13x13 DoG 필터 만들기



$Gaussian\ filter$

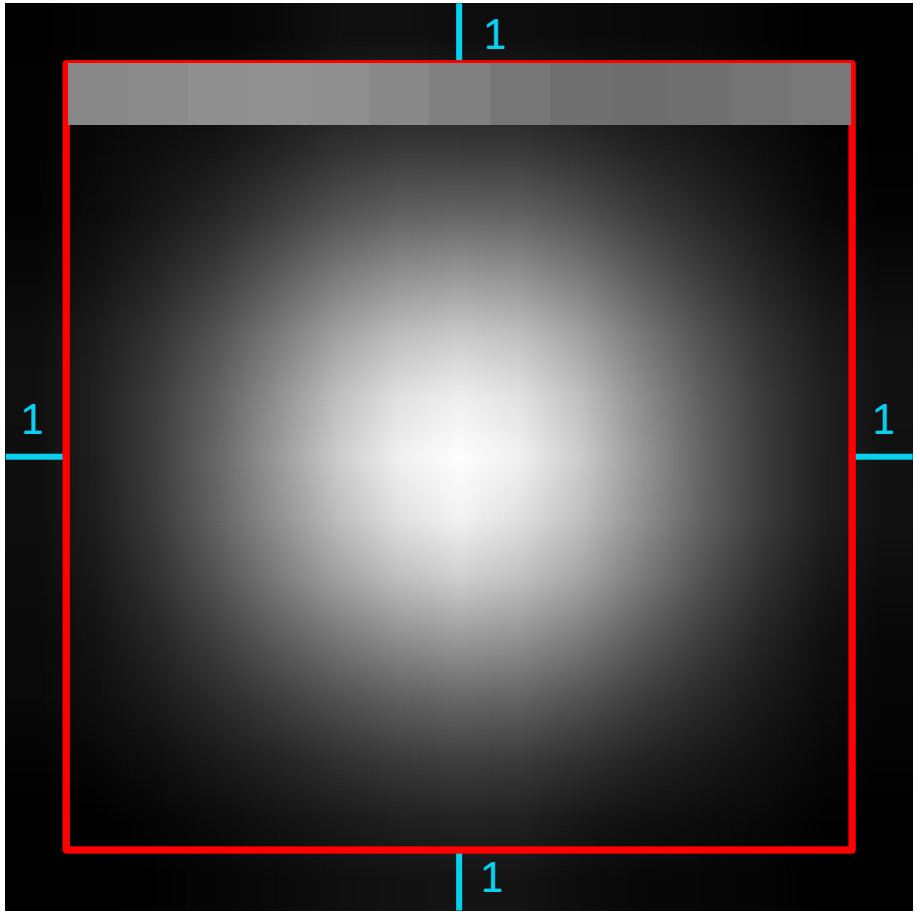
과제 DoG.py

- **Sigma값이 3인 13x13 DoG 필터 만들기**

$Gaussian\ filter$

과제 DoG.py

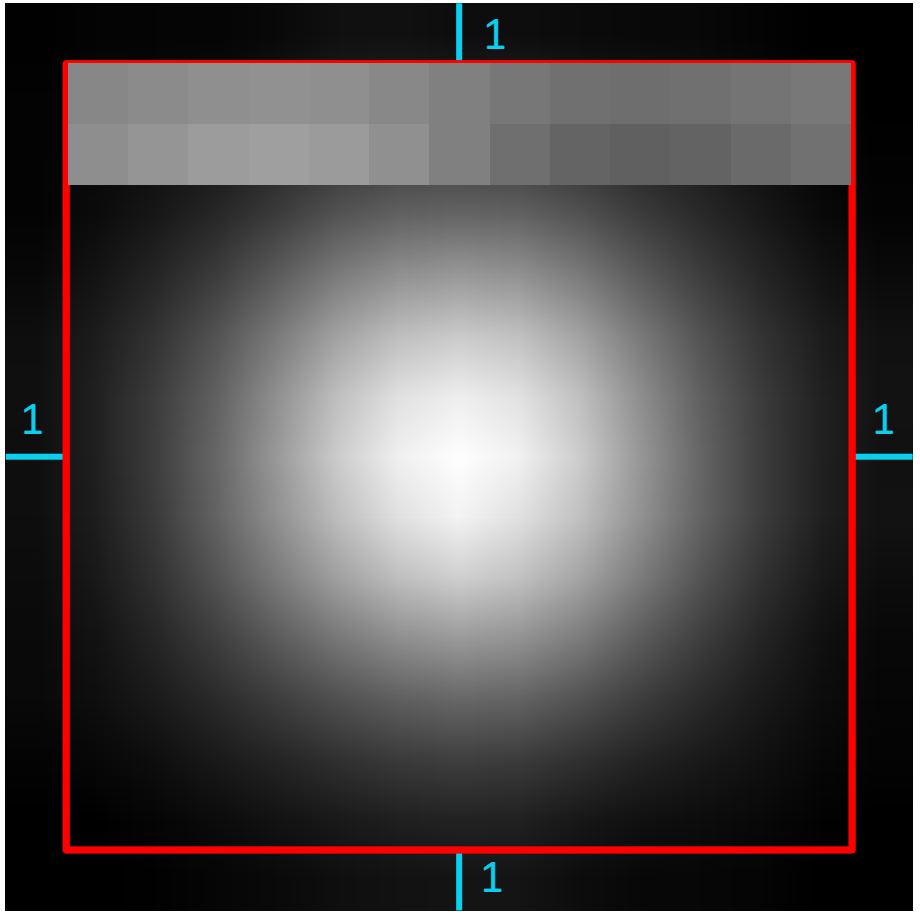
- **Sigma값이 3인 13x13 DoG 필터 만들기**



$Gaussian\ filter$

과제 DoG.py

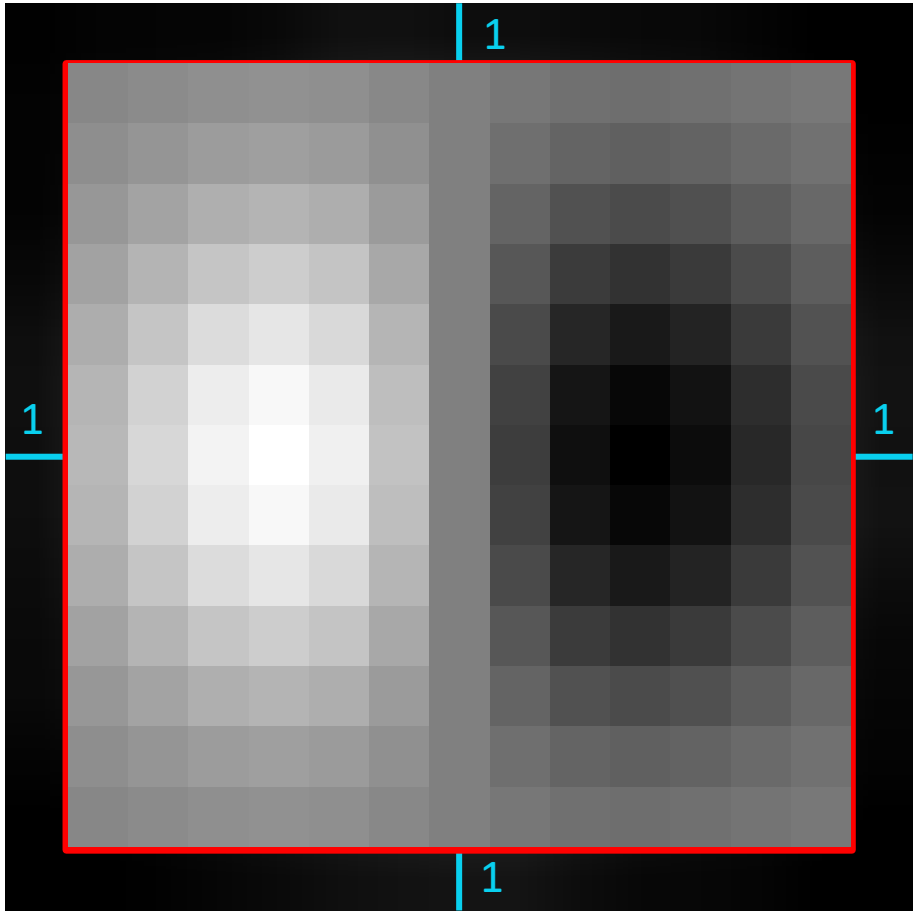
- **Sigma값이 3인 13x13 DoG 필터 만들기**



$Gaussian\ filter$

과제 DoG.py

- Sigma값이 3인 13x13 DoG 필터 만들기

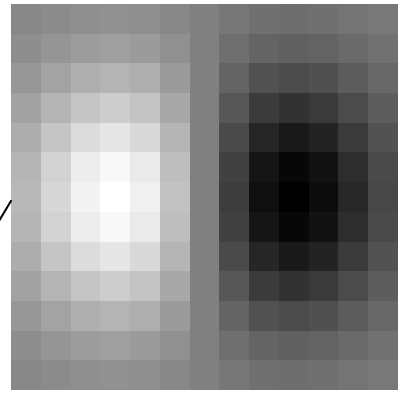


$Gaussian\ filter$

과제 DoG.py

f
(image)

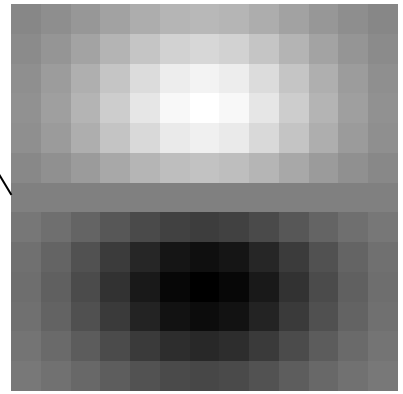
*



$DoG\ filter\ x$

f_x

*



$DoG\ filter\ y$

f_y

$\sqrt{{f_x}^2 + {f_y}^2}$

M
(DoG_filtering)

Overflow 막기 위해
np.float32로 이미지 불러오기

과제 DoG.py

- **noise_Lena.png**

Original

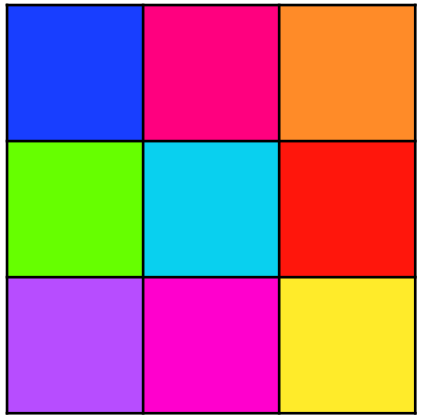
Sobel

DoG (Equation)

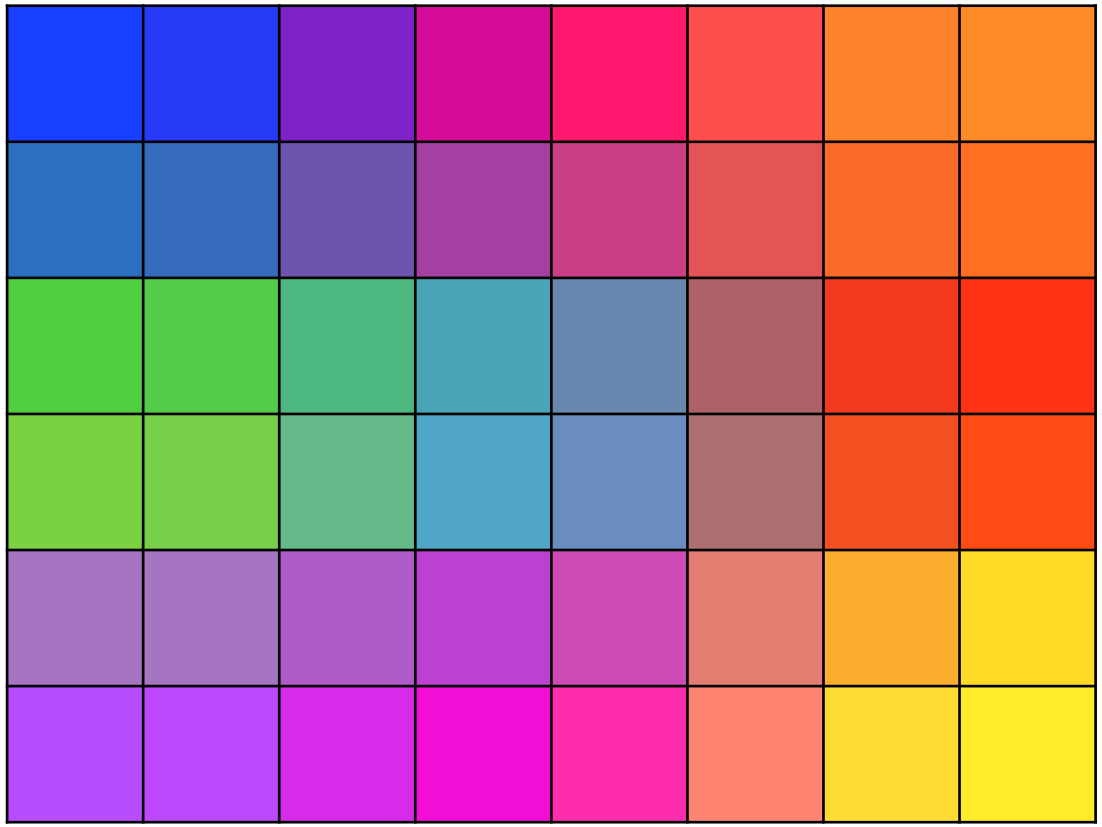
DoG (Filtering)

과제 my_resize_1by1.py

- **Resize 3x3 → 6x8**
  - Image upsampling



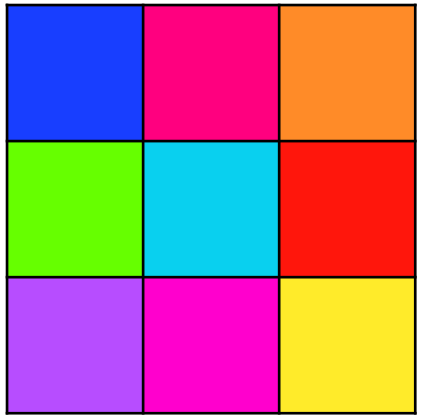
3x3 image



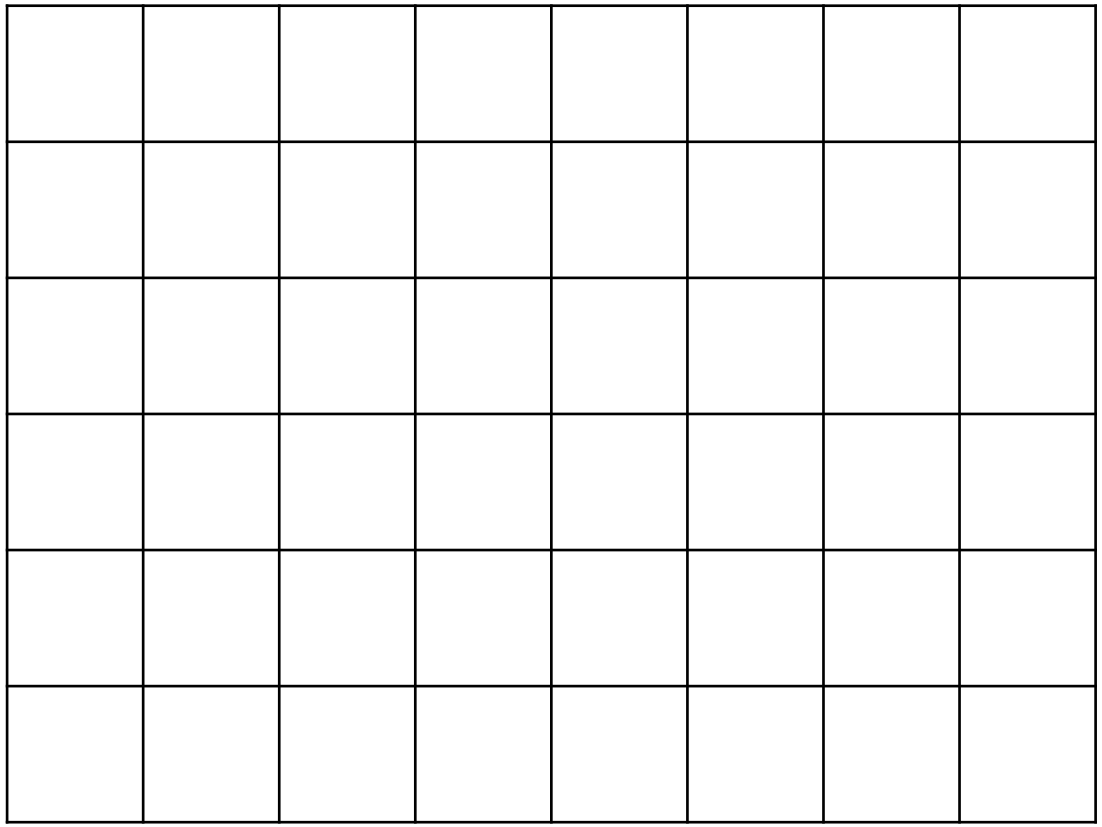
6x8 image

과제 my_resize_1by1.py

- **Resize 3x3 → 6x8**
  1. Create new image
     - (6, 8)의 빈 배열 만들기 (0으로 채우기)



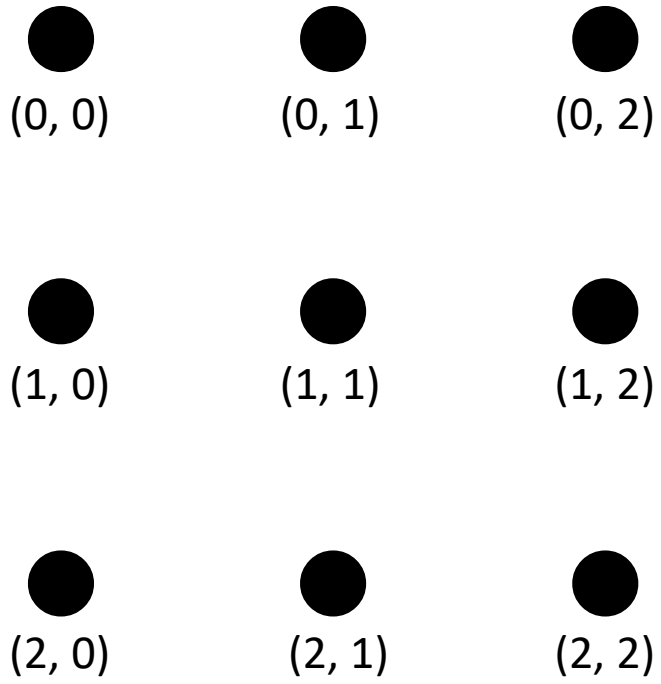
3x3 image
(Old)



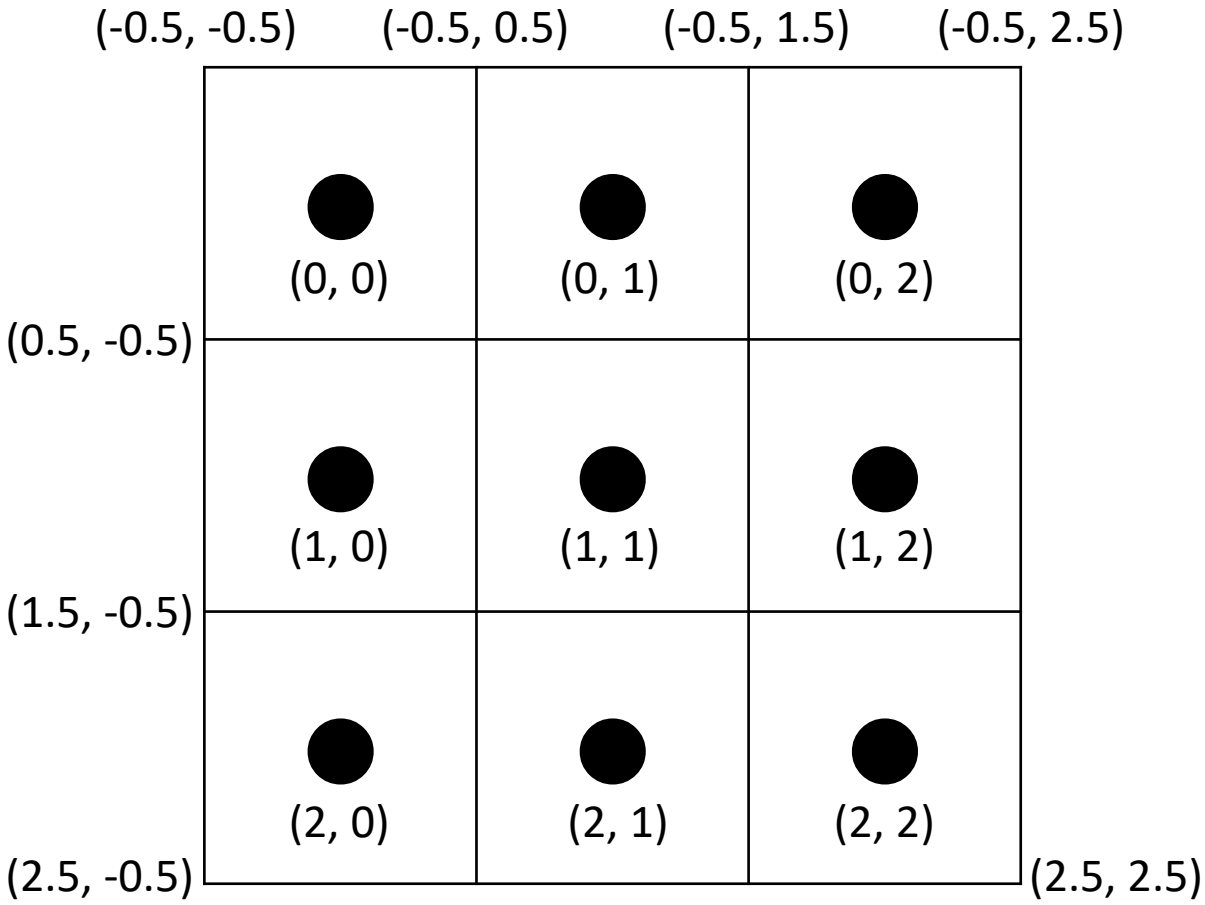
6x8 image
(New)

과제 my_resize_1by1.py

- **Resize 3x3 → 6x8**
  2. Match up coordinates
     - 픽셀의 영역

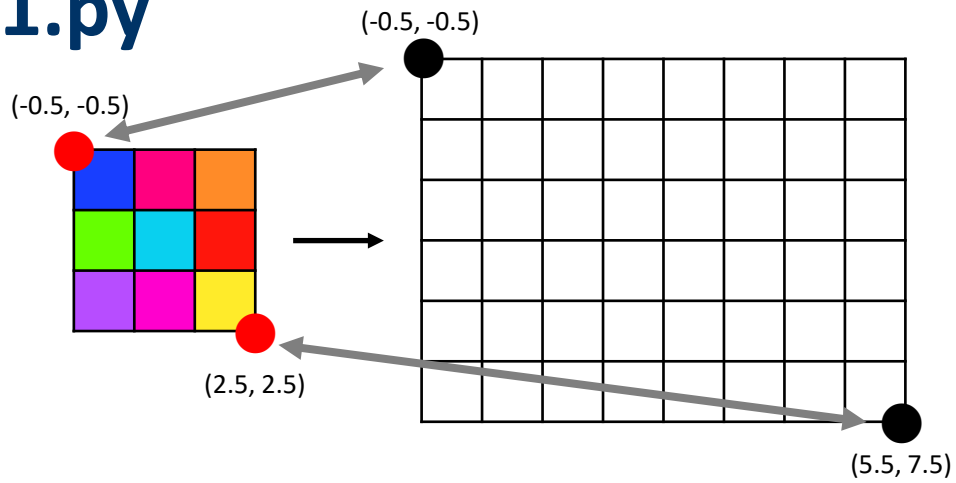


3x3 image (Point)



3x3 image (1x1 area)

과제 my_resize_1by1.py



- **Resize 3x3 → 6x8**

2. Match up coordinates (Pixel: 1x1 Area)
   - $y_{old} = a_y y_{new} + b_y$
   - $x_{old} = a_x y_{new} + b_x$
     - Old image 좌측 상단 좌표 ↔ New image 좌측 상단 좌표

       $(-0.5, -0.5) \quad \leftrightarrow \quad (-0.5, -0.5)$
     - Old image 우측 하단 좌표 ↔ New image 우측 하단 좌표

       $(2.5, 2.5) \quad \leftrightarrow \quad (5.5, 7.5)$

$$\begin{cases} -0.5 \cdot a_y + b_y = -0.5 \\ 5.5 \cdot a_y + b_y = 2.5 \end{cases}, \quad a_y = \frac{1}{2}, \quad b_y = -\frac{1}{4}, \quad \therefore y_{old} = \frac{1}{2} y_{new} - \frac{1}{4}$$

$$\begin{cases} -0.5 \cdot a_x + b_x = -0.5 \\ 7.5 \cdot a_x + b_x = 2.5 \end{cases}, \quad a_x = \frac{3}{8}, \quad b_x = -\frac{5}{16}, \quad \therefore x_{old} = \frac{3}{8} x_{new} - \frac{5}{16}$$

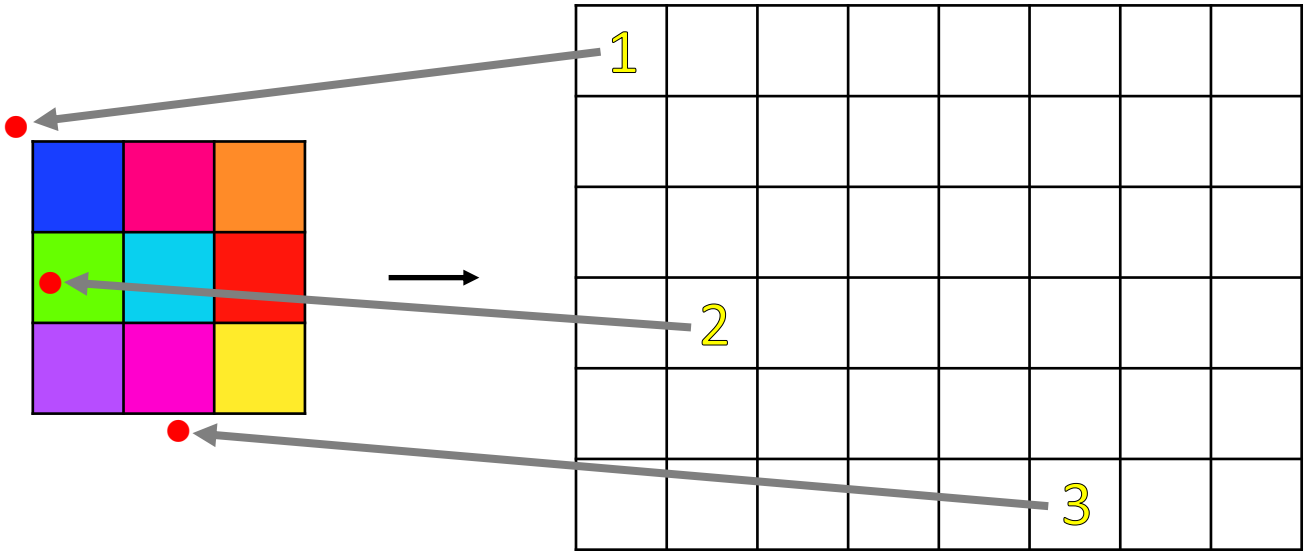
과제 my_resize_1by1.py

- **Resize 3x3 → 6x8**

  3. Iterate over new points

     - $y_{old} = \frac{1}{2} y_{new} - \frac{1}{4}$
     - $x_{old} = \frac{3}{8} x_{new} - \frac{5}{16}$
       - Mapping new coordinate to old coordinate

         1. $(0, 0) \rightarrow (-\frac{1}{4}, -\frac{5}{16})$
         2. $(4, 1) \rightarrow (\frac{7}{4}, \frac{1}{16})$
         3. $(5, 5) \rightarrow (\frac{9}{4}, \frac{25}{16})$



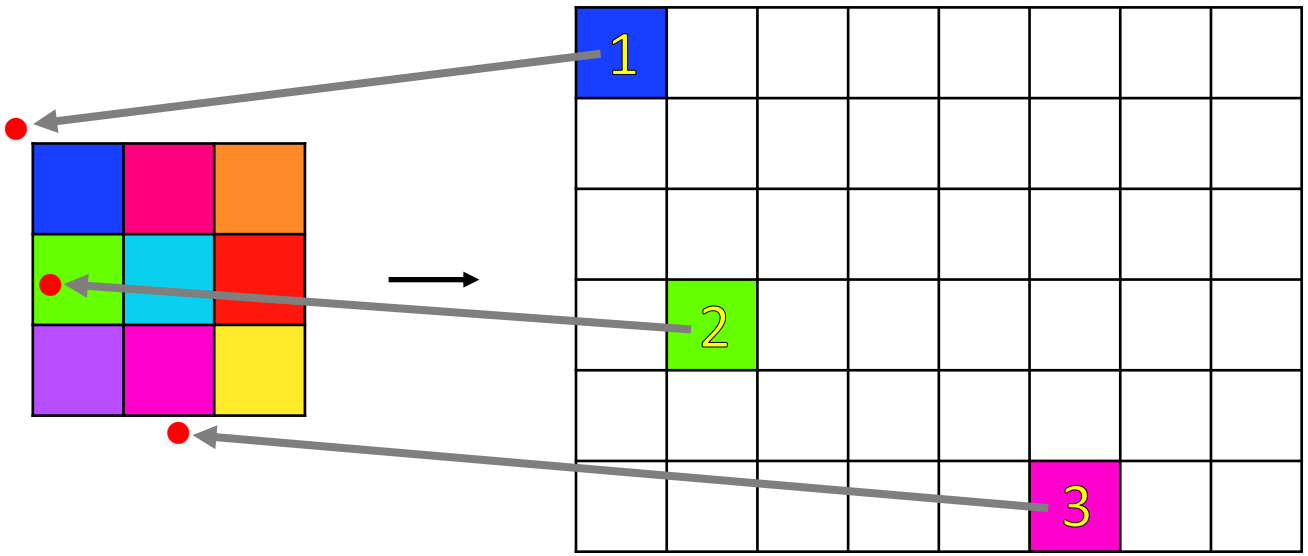
과제 my_resize_1by1.py

- **Resize 3x3 → 6x8**

  3. Iterate over new points

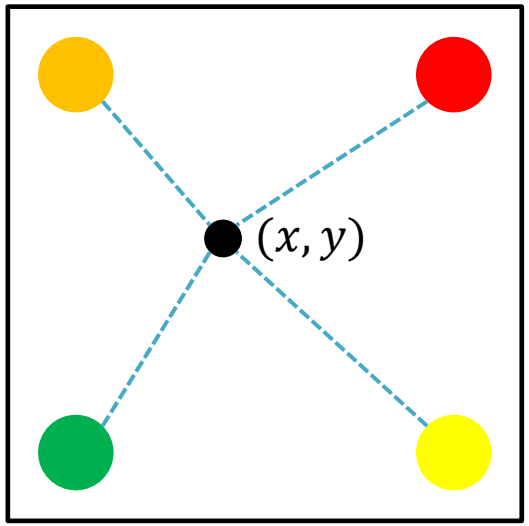
     - $y_{old} = \frac{1}{2} y_{new} - \frac{1}{4}$
     - $x_{old} = \frac{3}{8} x_{new} - \frac{5}{16}$
       - Mapping new coordinate to old coordinate

         1. $(0, 0) \rightarrow (-\frac{1}{4}, -\frac{5}{16})$
         2. $(4, 1) \rightarrow (\frac{7}{4}, \frac{1}{16})$
         3. $(5, 5) \rightarrow (\frac{9}{4}, \frac{25}{16})$

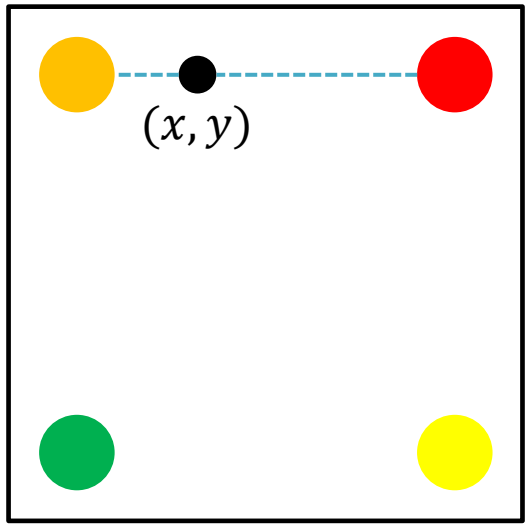


과제 my_resize_1by1.py

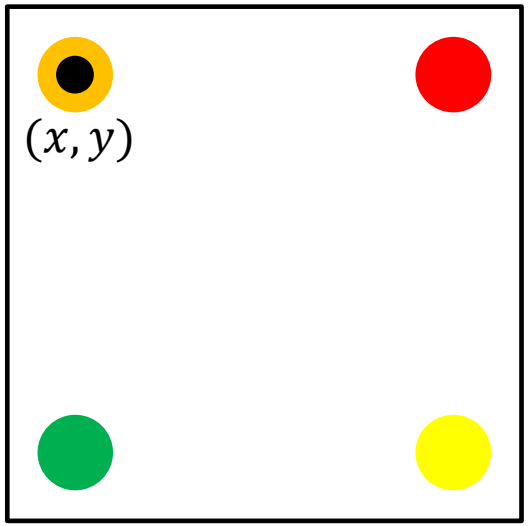
- **Resize 3x3 → 6x8**
  3. Iterate over new points
     - 4 Case



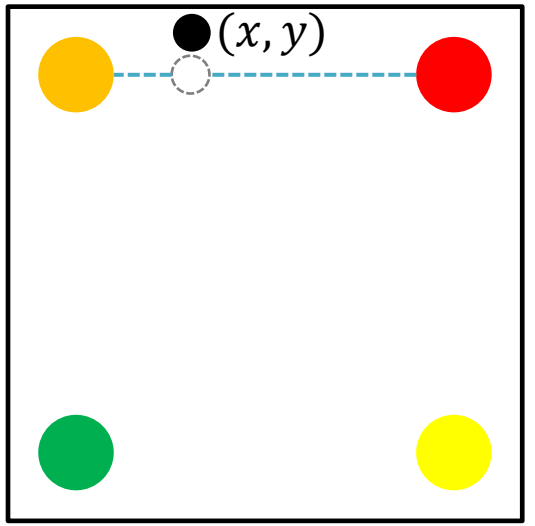
1. 네 점 사이
(float, float)



2. 두 점 사이
(float, int)
or
(int, float)

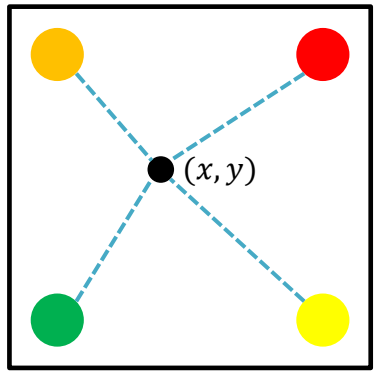


3. 점과 일치
(int, int)



4. Out of bounds
예시(float, float)
→
(float, int)

과제 my_resize_1by1.py

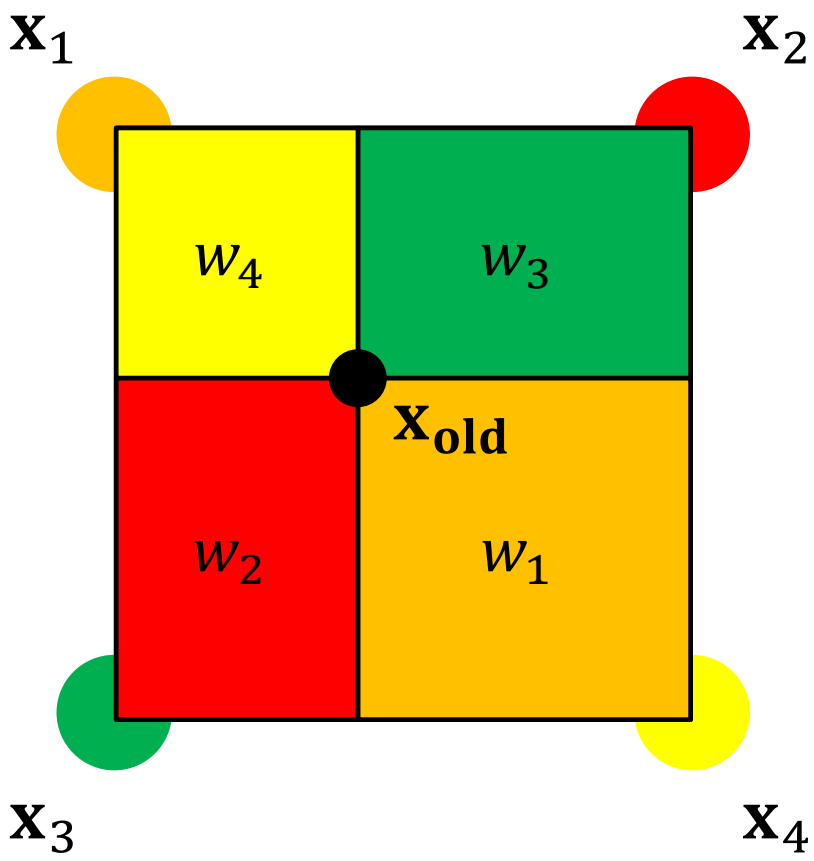


1. 네 점 사이

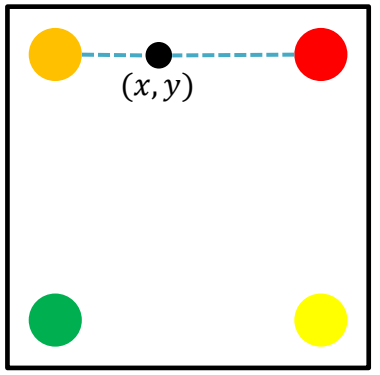
- **Resize 3x3 → 6x8**

4. Case 1 – Bilinear interpolation

- $f_{new}(\mathbf{x}_{new}) = \sum_{i=1}^{4} w_i f_{old}(\mathbf{x}_i)$
  - $\mathbf{x}_{old} = (x, y)$
  - $\mathbf{x}_1 = (\lfloor x \rfloor, \lfloor y \rfloor)$
  - $\mathbf{x}_2 = (\lfloor x + 1 \rfloor, \lfloor y \rfloor)$
  - $\mathbf{x}_3 = (\lfloor x \rfloor, \lfloor y + 1 \rfloor)$
  - $\mathbf{x}_4 = (\lfloor x + 1 \rfloor, \lfloor y + 1 \rfloor)$
  - $w_1 = (\lfloor x + 1 \rfloor - x)(\lfloor y + 1 \rfloor - y)$
  - $w_2 = (x - \lfloor x \rfloor)(\lfloor y + 1 \rfloor - y)$
  - $w_3 = (x - \lfloor x \rfloor)(\lfloor y + 1 \rfloor - y)$
  - $w_4 = (x - \lfloor x \rfloor)(y - \lfloor y \rfloor)$



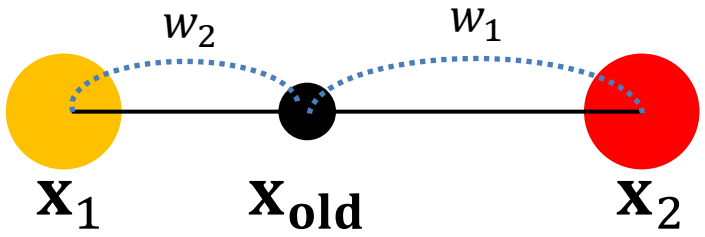
과제 my_resize_1by1.py



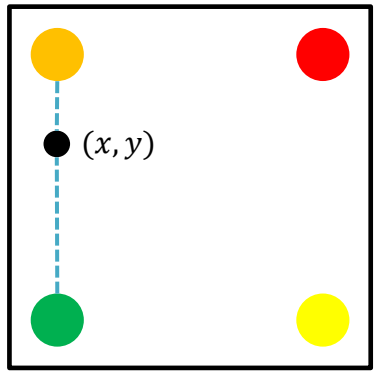
2. 두 점 사이
(float, int)

- **Resize 3x3 → 6x8**

4. Case 2 – Linear interpolation
   - $f_{new}(\mathbf{x}_{new}) = \sum_{i=1}^{2} w_i f_{old}(\mathbf{x}_i)$
     - $\mathbf{x}_1 = (\lfloor x \rfloor, \lfloor y \rfloor)$
     - $\mathbf{x}_2 = (\lfloor x + 1 \rfloor, \lfloor y \rfloor)$
     - $w_1 = (\lfloor x + 1 \rfloor - x)$
     - $w_2 = (x - \lfloor x \rfloor)$



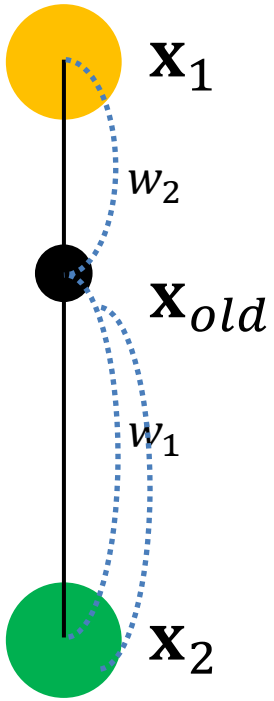
과제 my_resize_1by1.py



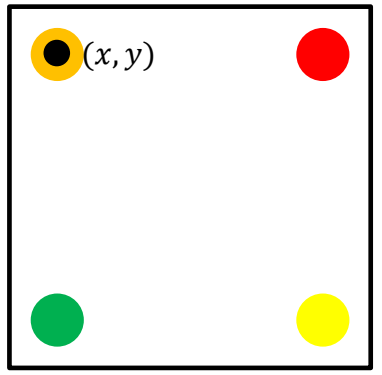
2. 두 점 사이
(float, int)

- **Resize 3x3 → 6x8**

4. Case 2 – Linear interpolation
   - $f_{new}(\mathbf{x}_{new}) = \sum_{i=1}^{2} w_i f_{old}(\mathbf{x}_i)$
     - $\mathbf{x}_1 = (\lfloor x \rfloor, \lfloor y \rfloor)$
     - $\mathbf{x}_2 = (\lfloor x \rfloor, \lfloor y + 1 \rfloor)$
     - $w_1 = (\lfloor y + 1 \rfloor - y)$
     - $w_2 = (y - \lfloor y \rfloor)$



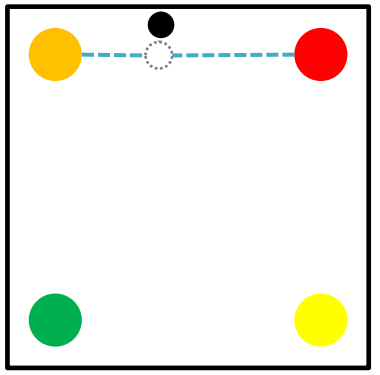
과제 my_resize_1by1.py



3. 점과 일치
(int, int)

- **Resize 3x3 → 6x8**
  4. Case 3
     - $f_{new}(\mathbf{x}_{new}) = f_{old}(\mathbf{x}_{\mathrm{old}})$

과제 my_resize_1by1.py



4. Out of bounds
(int, int)

- **Resize 3x3 → 6x8**

4. Case 4

1) y 좌표가 float:

1: $y = max(y, 0)$

2: $y = min(y, h - 1)$

2) x 좌표가 float:

1: $y = min(y, h - 1)$

2: $f(x) = f(min(x, w - 1))$

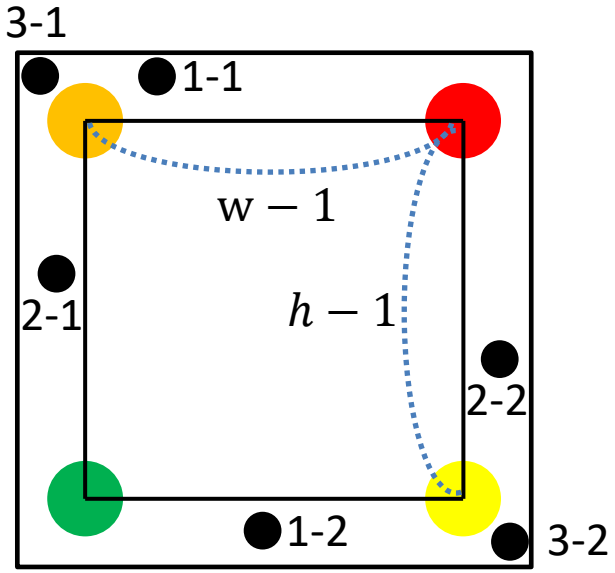
3) x 좌표와 y 좌표 모두 float:

1: $f(x) = f(max(x, 0))$

$f(y) = f(max(y, 0))$

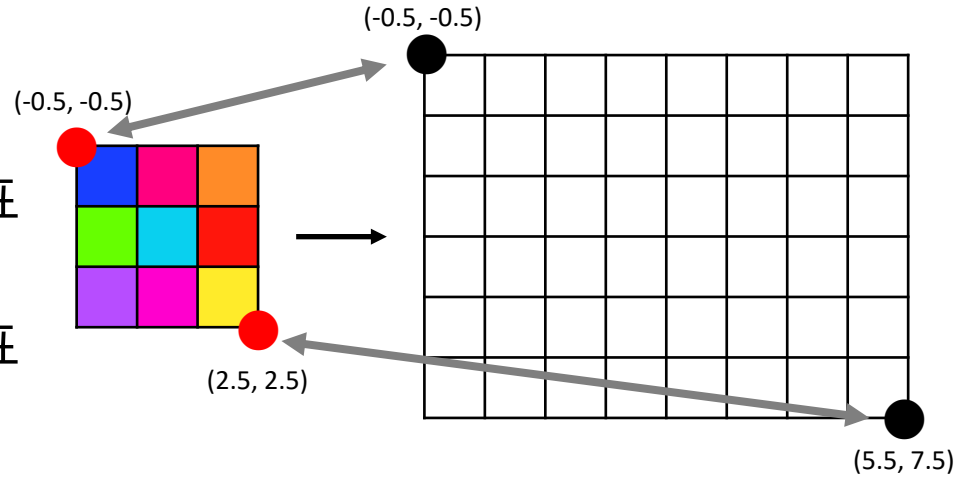
2: $f(x) = f(min(x, w - 1))$

$f(y) = f(max(y, h - 1))$



과제 my_resize_1by1.py

- **match_up_coordinates()**
  - $y_{old} = a_y y_{new} + b_y$
  - $x_{old} = a_x x_{new} + b_x$
    - Old image 좌측 상단 좌표 ↔ New image 좌측 상단 좌표
      - $(-0.5, -0.5) \leftrightarrow (-0.5, -0.5)$
    - Old image 우측 하단 좌표 ↔ New image 우측 하단 좌표
      - $(h_{old} - 0.5, w_{old} - 0.5) \leftrightarrow (h_{new} - 0.5, w_{new} - 0.5)$



$$\begin{cases} -0.5a_x + b_x = -0.5 \\ (h_{new} - 0.5)a_x + b_x = h_{old} - 0.5 \end{cases}$$

$$a_y = \frac{h_{old}}{h_{new}}, \qquad b_y = 0.5(1 - a_y)$$

$$a_x = \frac{w_{old}}{w_{new}}, \qquad b_x = 0.5(1 - a_x)$$

과제

- **보고서**
  - 내용
    - 학과, 학번, 이름
    - 구현 코드: 구현한 코드에 대한 간단한 설명
    - 이미지: 언급한 이미지 모두 첨부
    - 느낀 점: 구현 결과를 보고 느낀 점, 혹은 어려운 점 등
    - 과제 난이도: 개인적으로 느낀 난이도 및 이유(과제가 쉽다, 어렵다 등)
  - .pdf 파일로 제출(이외의 파일 형식일 경우 감점)
  - 보고서 명
    - [IP]20xxxxxxx_이름_x주차_과제.pdf

과제

- **과제 안내**
  - 채점 기준
    - 구현을 못하거나 잘못 구현한 경우
    - 보고서 내용이 빠진 경우
    - 다른 사람의 코드 copy 적발시 보여준 사람, copy한 사람 둘 다 0점
    - 내장 함수 사용시 감점(내장 함수를 사용해도 된다고 한 것 제외)
  - 제출 파일
    - 아래의 파일을 압축해서 [IP]20XXXXXXX_이름_X주차_과제.zip 으로 제출
      - .py 파일
      - .pdf 보고서 파일
  - 제출 기한
    - 2024년 4월 25일 23시 59분까지

Q & A