Image Processing

실습 7주차

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)

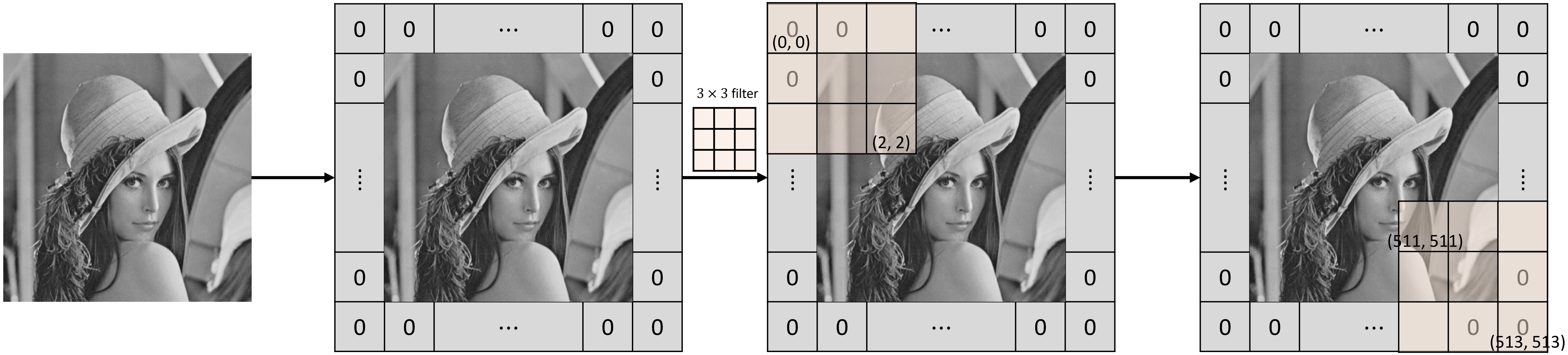
- **TA 연락처**
  - 공대 5호관 531호 컴퓨터비전 연구실
  - 과제 질문은 [IP]를 제목에 붙여 메일로 주세요.
  - 00반
    - 안준혁
    - ajh99345@gmail.com
  - 01반
    - 신동헌
    - doghon85@naver.com

목차

- **4주차 과제1 리뷰**
- **4주차 과제2 리뷰**
- **과제**
  - Canny edge detection

4주차 과제 1 리뷰

- **my_filtering(src, kernel, pad_type) 구현**
  - (512, 512)인 이미지를 3 × 3 filter를 사용해서 filtering 진행한다고 가정
    - padding 후 이미지 형태는 (514, 514)



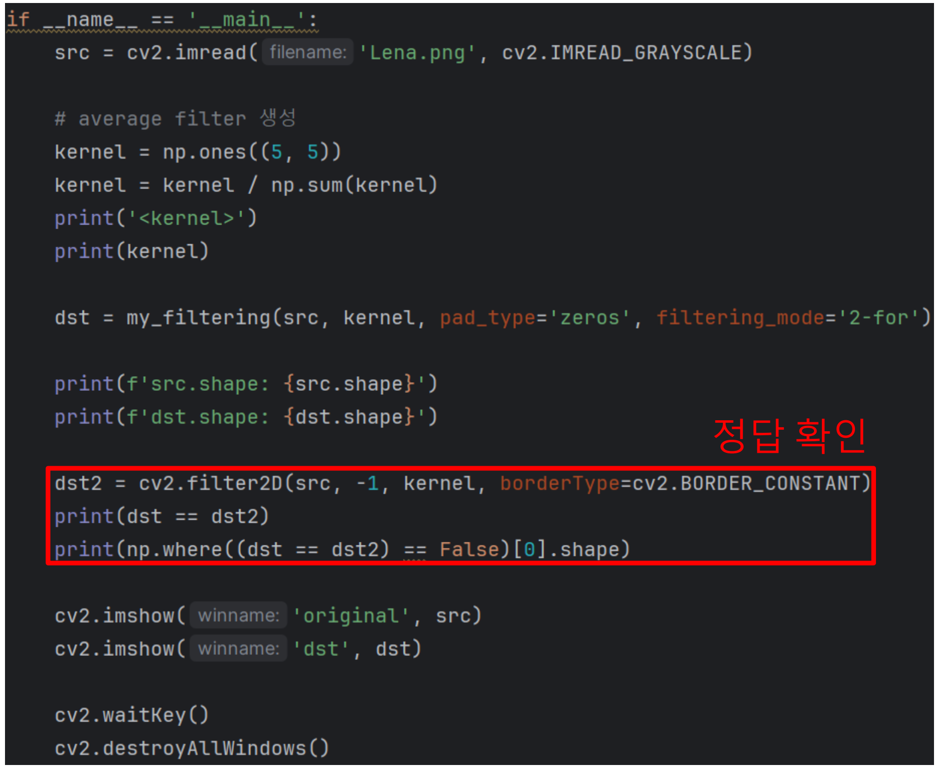
src

Padded image

Filtering

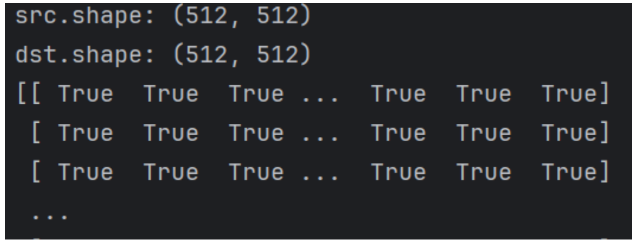
4주차 과제 1 리뷰

- my_filtering(src, kernel, pad_type) 구현

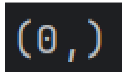


```
if __name__ == '__main__':
    src = cv2.imread( filename: 'Lena.png', cv2.IMREAD_GRAYSCALE)

    # average filter 생성
    kernel = np.ones((5, 5))
    kernel = kernel / np.sum(kernel)
    print('<kernel>')
    print(kernel)

    dst = my_filtering(src, kernel, pad_type='zeros', filtering_mode='2-for')

    print(f'src.shape: {src.shape}')
    print(f'dst.shape: {dst.shape}')

    dst2 = cv2.filter2D(src, -1, kernel, borderType=cv2.BORDER_CONSTANT)
    print(dst == dst2)
    print(np.where((dst == dst2) == False)[0].shape)

    cv2.imshow( winname: 'original', src)
    cv2.imshow( winname: 'dst', dst)

    cv2.waitKey()
    cv2.destroyAllWindows()
```

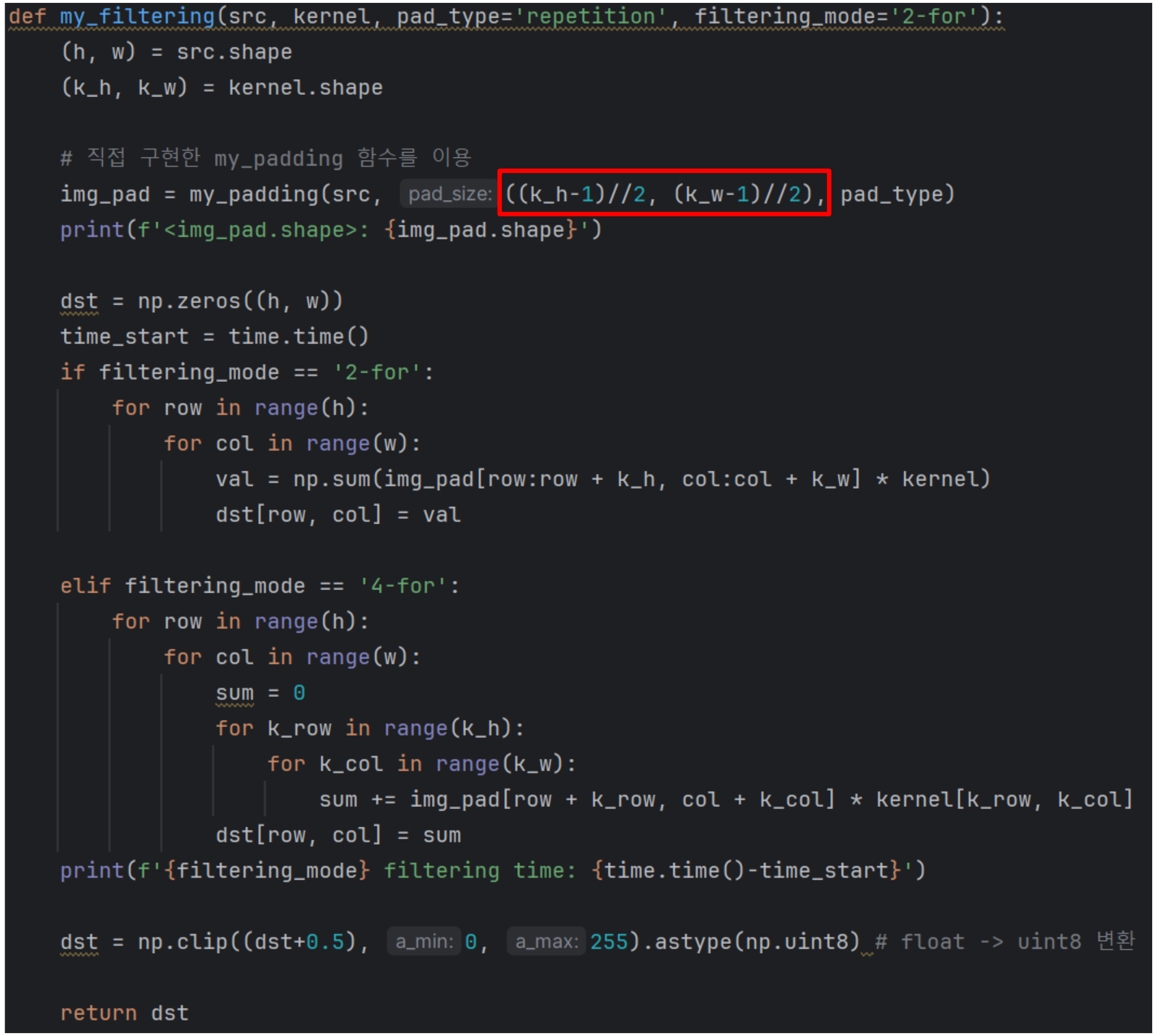
정답 확인



```
src.shape: (512, 512)
dst.shape: (512, 512)
[[ True  True  True ...  True  True  True]
 [ True  True  True ...  True  True  True]
 [ True  True  True ...  True  True  True]
 ...
```



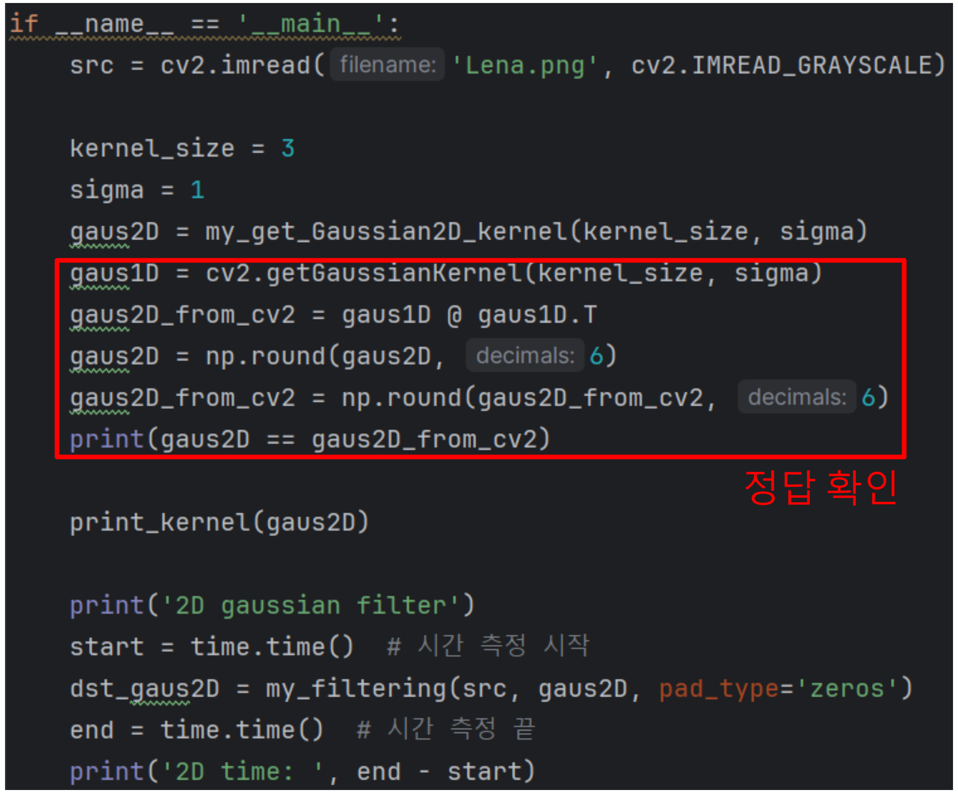
```
(0,)
```



```
def my_filtering(src, kernel, pad_type='repetition', filtering_mode='2-for'):
    (h, w) = src.shape
    (k_h, k_w) = kernel.shape

    # 직접 구현한 my_padding 함수를 이용
    img_pad = my_padding(src,  pad_size: ((k_h-1)//2, (k_w-1)//2), pad_type)
    print(f'<img_pad.shape>: {img_pad.shape}')

    dst = np.zeros((h, w))
    time_start = time.time()
    if filtering_mode == '2-for':
        for row in range(h):
            for col in range(w):
                val = np.sum(img_pad[row:row + k_h, col:col + k_w] * kernel)
                dst[row, col] = val

    elif filtering_mode == '4-for':
        for row in range(h):
            for col in range(w):
                sum = 0
                for k_row in range(k_h):
                    for k_col in range(k_w):
                        sum += img_pad[row + k_row, col + k_col] * kernel[k_row, k_col]
                dst[row, col] = sum
    print(f'{filtering_mode} filtering time: {time.time()-time_start}')

    dst = np.clip((dst+0.5),  a_min: 0,  a_max: 255).astype(np.uint8) # float -> uint8 변환

    return dst
```

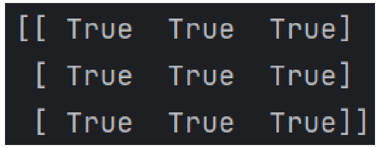
4주차 과제 2 리뷰

- **my_get_Gaussian2D_kernel(ksize, sigma) 구현**
  - Filter 크기와 σ를 입력받아 Gaussian 2d filter를 반환하는 함수 구현
  - my_filtering 함수는 과제 1과 동일

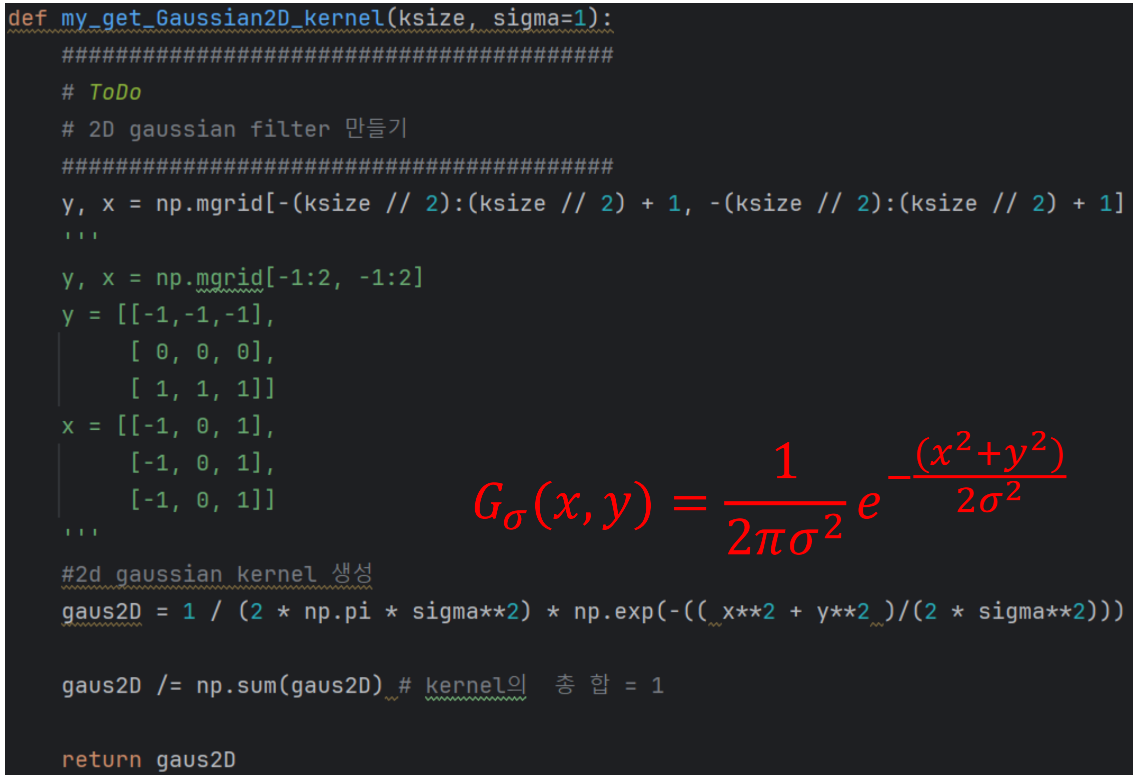


```
if __name__ == '__main__':
    src = cv2.imread( filename: 'Lena.png', cv2.IMREAD_GRAYSCALE)

    kernel_size = 3
    sigma = 1
    gaus2D = my_get_Gaussian2D_kernel(kernel_size, sigma)
    gaus1D = cv2.getGaussianKernel(kernel_size, sigma)
    gaus2D_from_cv2 = gaus1D @ gaus1D.T
    gaus2D = np.round(gaus2D,  decimals: 6)
    gaus2D_from_cv2 = np.round(gaus2D_from_cv2,  decimals: 6)
    print(gaus2D == gaus2D_from_cv2)

    print_kernel(gaus2D)

    print('2D gaussian filter')
    start = time.time()  # 시간 측정 시작
    dst_gaus2D = my_filtering(src, gaus2D, pad_type='zeros')
    end = time.time()  # 시간 측정 끝
    print('2D time: ', end - start)
```

정답 확인



```
[[ True  True  True]
 [ True  True  True]
 [ True  True  True]]
```

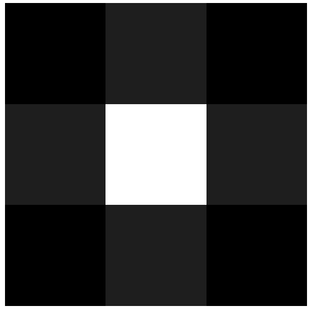


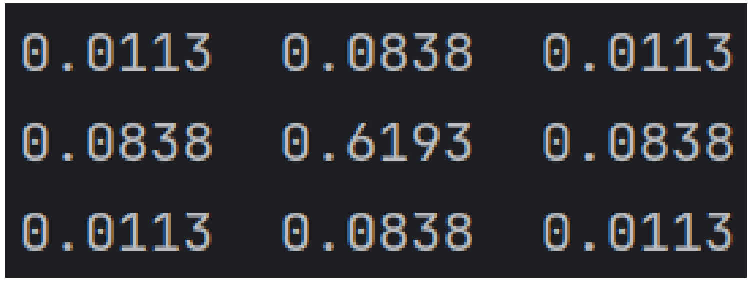
```
def my_get_Gaussian2D_kernel(ksize, sigma=1):
    ##########################################
    # ToDo
    # 2D gaussian filter 만들기
    ##########################################
    y, x = np.mgrid[-(ksize // 2):(ksize // 2) + 1, -(ksize // 2):(ksize // 2) + 1]
    '''
    y, x = np.mgrid[-1:2, -1:2]
    y = [[-1,-1,-1],
         [ 0, 0, 0],
         [ 1, 1, 1]]
    x = [[-1, 0, 1],
         [-1, 0, 1],
         [-1, 0, 1]]
    '''
    #2d gaussian kernel 생성
    gaus2D = 1 / (2 * np.pi * sigma**2) * np.exp(-(( x**2 + y**2 )/(2 * sigma**2)))

    gaus2D /= np.sum(gaus2D) # kernel의  총 합 = 1

    return gaus2D
```

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

4주차 과제 2 리뷰

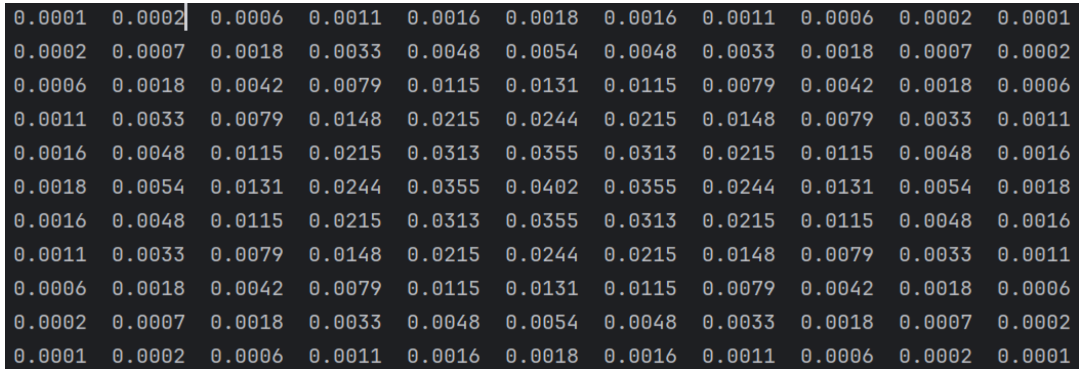
- σ에 따른 kernel 크기 선정



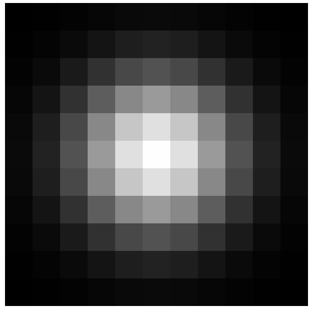


```
0.0113   0.0838   0.0113
0.0838   0.6193   0.0838
0.0113   0.0838   0.0113
```

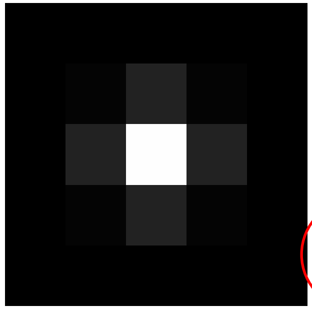
$\sigma = 0.5$, kernel 크기 3인 Gaussian filter

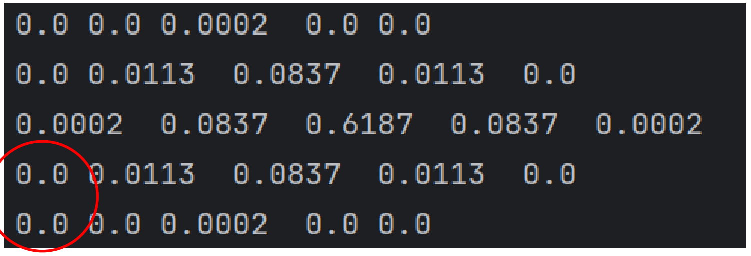


```
0.0001  0.0002  0.0006  0.0011  0.0016  0.0018  0.0016  0.0011  0.0006  0.0002  0.0001
0.0002  0.0007  0.0018  0.0033  0.0048  0.0054  0.0048  0.0033  0.0018  0.0007  0.0002
0.0006  0.0018  0.0042  0.0079  0.0115  0.0131  0.0115  0.0079  0.0042  0.0018  0.0006
0.0011  0.0033  0.0079  0.0148  0.0215  0.0244  0.0215  0.0148  0.0079  0.0033  0.0011
0.0016  0.0048  0.0115  0.0215  0.0313  0.0355  0.0313  0.0215  0.0115  0.0048  0.0016
0.0018  0.0054  0.0131  0.0244  0.0355  0.0402  0.0355  0.0244  0.0131  0.0054  0.0018
0.0016  0.0048  0.0115  0.0215  0.0313  0.0355  0.0313  0.0215  0.0115  0.0048  0.0016
0.0011  0.0033  0.0079  0.0148  0.0215  0.0244  0.0215  0.0148  0.0079  0.0033  0.0011
0.0006  0.0018  0.0042  0.0079  0.0115  0.0131  0.0115  0.0079  0.0042  0.0018  0.0006
0.0002  0.0007  0.0018  0.0033  0.0048  0.0054  0.0048  0.0033  0.0018  0.0007  0.0002
0.0001  0.0002  0.0006  0.0011  0.0016  0.0018  0.0016  0.0011  0.0006  0.0002  0.0001
```



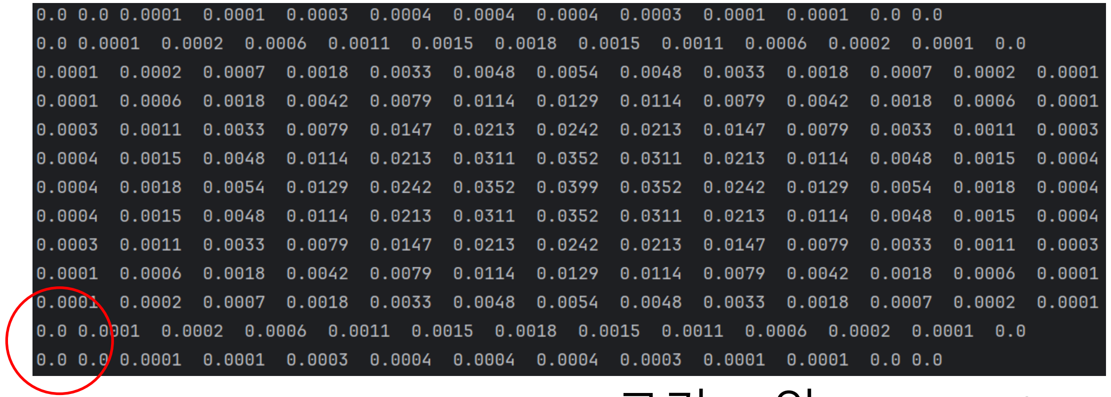
$\sigma = 2$, kernel 크기 11인 Gaussian filter



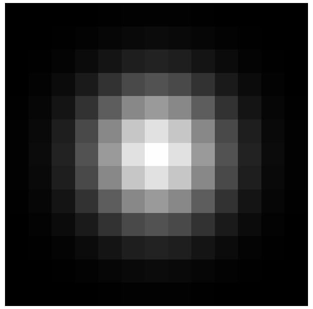


```
0.0 0.0 0.0002  0.0 0.0
0.0 0.0113  0.0837  0.0113  0.0
0.0002  0.0837  0.6187  0.0837  0.0002
0.0 0.0113  0.0837  0.0113  0.0
0.0 0.0 0.0002  0.0 0.0
```

$\sigma = 0.5$, kernel 크기 5인 Gaussian filter



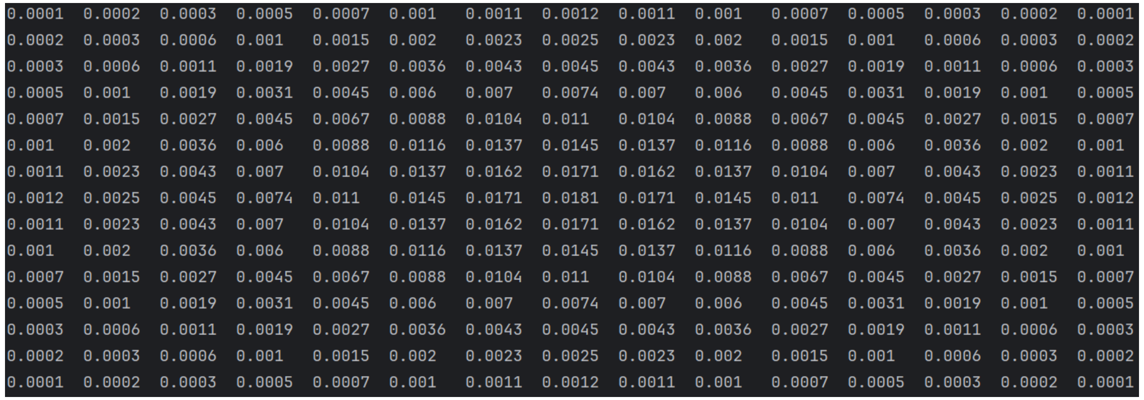
```
0.0 0.0 0.0001  0.0001  0.0003  0.0004  0.0004  0.0004  0.0003  0.0001  0.0001  0.0 0.0
0.0 0.0001  0.0002  0.0006  0.0011  0.0015  0.0018  0.0015  0.0011  0.0006  0.0002  0.0001  0.0
0.0001  0.0002  0.0007  0.0018  0.0033  0.0048  0.0054  0.0048  0.0033  0.0018  0.0007  0.0002  0.0001
0.0001  0.0006  0.0018  0.0042  0.0079  0.0114  0.0129  0.0114  0.0079  0.0042  0.0018  0.0006  0.0001
0.0003  0.0011  0.0033  0.0079  0.0147  0.0213  0.0242  0.0213  0.0147  0.0079  0.0033  0.0011  0.0003
0.0004  0.0015  0.0048  0.0114  0.0213  0.0311  0.0352  0.0311  0.0213  0.0114  0.0048  0.0015  0.0004
0.0004  0.0018  0.0054  0.0129  0.0242  0.0352  0.0399  0.0352  0.0242  0.0129  0.0054  0.0018  0.0004
0.0004  0.0015  0.0048  0.0114  0.0213  0.0311  0.0352  0.0311  0.0213  0.0114  0.0048  0.0015  0.0004
0.0003  0.0011  0.0033  0.0079  0.0147  0.0213  0.0242  0.0213  0.0147  0.0079  0.0033  0.0011  0.0003
0.0001  0.0006  0.0018  0.0042  0.0079  0.0114  0.0129  0.0114  0.0079  0.0042  0.0018  0.0006  0.0001
0.0001  0.0002  0.0007  0.0018  0.0033  0.0048  0.0054  0.0048  0.0033  0.0018  0.0007  0.0002  0.0001
0.0 0.0001  0.0002  0.0006  0.0011  0.0015  0.0018  0.0015  0.0011  0.0006  0.0002  0.0001  0.0
0.0 0.0 0.0001  0.0001  0.0003  0.0004  0.0004  0.0004  0.0003  0.0001  0.0001  0.0 0.0
```



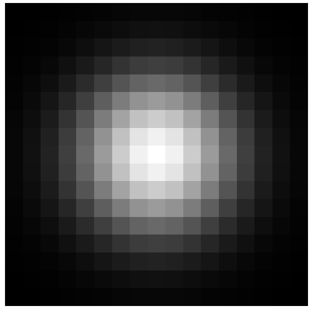
$\sigma = 2$, kernel 크기 13인 Gaussian filter

4주차 과제 2 리뷰

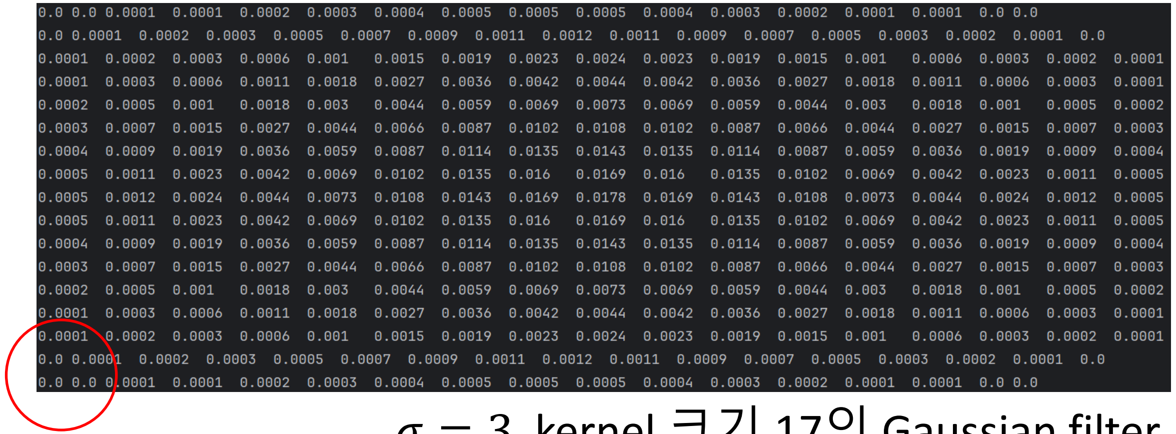
- σ에 따른 kernel 크기 선정



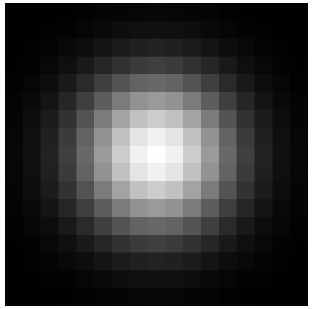
```
0.0001  0.0002  0.0003  0.0005  0.0007  0.001   0.0011  0.0012  0.0011  0.001   0.0007  0.0005  0.0003  0.0002  0.0001
0.0002  0.0003  0.0006  0.001   0.0015  0.002   0.0023  0.0025  0.0023  0.002   0.0015  0.001   0.0006  0.0003  0.0002
0.0003  0.0006  0.0011  0.0019  0.0027  0.0036  0.0043  0.0045  0.0043  0.0036  0.0027  0.0019  0.0011  0.0006  0.0003
0.0005  0.001   0.0019  0.0031  0.0045  0.006   0.007   0.0074  0.007   0.006   0.0045  0.0031  0.0019  0.001   0.0005
0.0007  0.0015  0.0027  0.0045  0.0067  0.0088  0.0104  0.011   0.0104  0.0088  0.0067  0.0045  0.0027  0.0015  0.0007
0.001   0.002   0.0036  0.006   0.0088  0.0116  0.0137  0.0145  0.0137  0.0116  0.0088  0.006   0.0036  0.002   0.001
0.0011  0.0023  0.0043  0.007   0.0104  0.0137  0.0162  0.0171  0.0162  0.0137  0.0104  0.007   0.0043  0.0023  0.0011
0.0012  0.0025  0.0045  0.0074  0.011   0.0145  0.0171  0.0181  0.0171  0.0145  0.011   0.0074  0.0045  0.0025  0.0012
0.0011  0.0023  0.0043  0.007   0.0104  0.0137  0.0162  0.0171  0.0162  0.0137  0.0104  0.007   0.0043  0.0023  0.0011
0.001   0.002   0.0036  0.006   0.0088  0.0116  0.0137  0.0145  0.0137  0.0116  0.0088  0.006   0.0036  0.002   0.001
0.0007  0.0015  0.0027  0.0045  0.0067  0.0088  0.0104  0.011   0.0104  0.0088  0.0067  0.0045  0.0027  0.0015  0.0007
0.0005  0.001   0.0019  0.0031  0.0045  0.006   0.007   0.0074  0.007   0.006   0.0045  0.0031  0.0019  0.001   0.0005
0.0003  0.0006  0.0011  0.0019  0.0027  0.0036  0.0043  0.0045  0.0043  0.0036  0.0027  0.0019  0.0011  0.0006  0.0003
0.0002  0.0003  0.0006  0.001   0.0015  0.002   0.0023  0.0025  0.0023  0.002   0.0015  0.001   0.0006  0.0003  0.0002
0.0001  0.0002  0.0003  0.0005  0.0007  0.001   0.0011  0.0012  0.0011  0.001   0.0007  0.0005  0.0003  0.0002  0.0001
```



$\sigma = 3$, kernel 크기 15인 Gaussian filter



```
0.0 0.0 0.0001  0.0001  0.0002  0.0003  0.0004  0.0005  0.0005  0.0005  0.0004  0.0003  0.0002  0.0001  0.0001  0.0 0.0
0.0 0.0001  0.0002  0.0003  0.0005  0.0007  0.0009  0.0011  0.0012  0.0011  0.0009  0.0007  0.0005  0.0003  0.0002  0.0001  0.0
0.0001  0.0002  0.0003  0.0006  0.001   0.0015  0.0019  0.0023  0.0024  0.0023  0.0019  0.0015  0.001   0.0006  0.0003  0.0002  0.0001
0.0001  0.0003  0.0006  0.0011  0.0018  0.0027  0.0036  0.0042  0.0044  0.0042  0.0036  0.0027  0.0018  0.0011  0.0006  0.0003  0.0001
0.0002  0.0005  0.001   0.0018  0.003   0.0044  0.0059  0.0069  0.0073  0.0069  0.0059  0.0044  0.003   0.0018  0.001   0.0005  0.0002
0.0003  0.0007  0.0015  0.0027  0.0044  0.0066  0.0087  0.0102  0.0108  0.0102  0.0087  0.0066  0.0044  0.0027  0.0015  0.0007  0.0003
0.0004  0.0009  0.0019  0.0036  0.0059  0.0087  0.0114  0.0135  0.0143  0.0135  0.0114  0.0087  0.0059  0.0036  0.0019  0.0009  0.0004
0.0005  0.0011  0.0023  0.0042  0.0069  0.0102  0.0135  0.016   0.0169  0.016   0.0135  0.0102  0.0069  0.0042  0.0023  0.0011  0.0005
0.0005  0.0012  0.0024  0.0044  0.0073  0.0108  0.0143  0.0169  0.0178  0.0169  0.0143  0.0108  0.0073  0.0044  0.0024  0.0012  0.0005
0.0005  0.0011  0.0023  0.0042  0.0069  0.0102  0.0135  0.016   0.0169  0.016   0.0135  0.0102  0.0069  0.0042  0.0023  0.0011  0.0005
0.0004  0.0009  0.0019  0.0036  0.0059  0.0087  0.0114  0.0135  0.0143  0.0135  0.0114  0.0087  0.0059  0.0036  0.0019  0.0009  0.0004
0.0003  0.0007  0.0015  0.0027  0.0044  0.0066  0.0087  0.0102  0.0108  0.0102  0.0087  0.0066  0.0044  0.0027  0.0015  0.0007  0.0003
0.0002  0.0005  0.001   0.0018  0.003   0.0044  0.0059  0.0069  0.0073  0.0069  0.0059  0.0044  0.003   0.0018  0.001   0.0005  0.0002
0.0001  0.0003  0.0006  0.0011  0.0018  0.0027  0.0036  0.0042  0.0044  0.0042  0.0036  0.0027  0.0018  0.0011  0.0006  0.0003  0.0001
0.0001  0.0002  0.0003  0.0006  0.001   0.0015  0.0019  0.0023  0.0024  0.0023  0.0019  0.0015  0.001   0.0006  0.0003  0.0002  0.0001
0.0 0.0001  0.0002  0.0003  0.0005  0.0007  0.0009  0.0011  0.0012  0.0011  0.0009  0.0007  0.0005  0.0003  0.0002  0.0001  0.0
0.0 0.0 0.0001  0.0001  0.0002  0.0003  0.0004  0.0005  0.0005  0.0005  0.0004  0.0003  0.0002  0.0001  0.0001  0.0 0.0
```

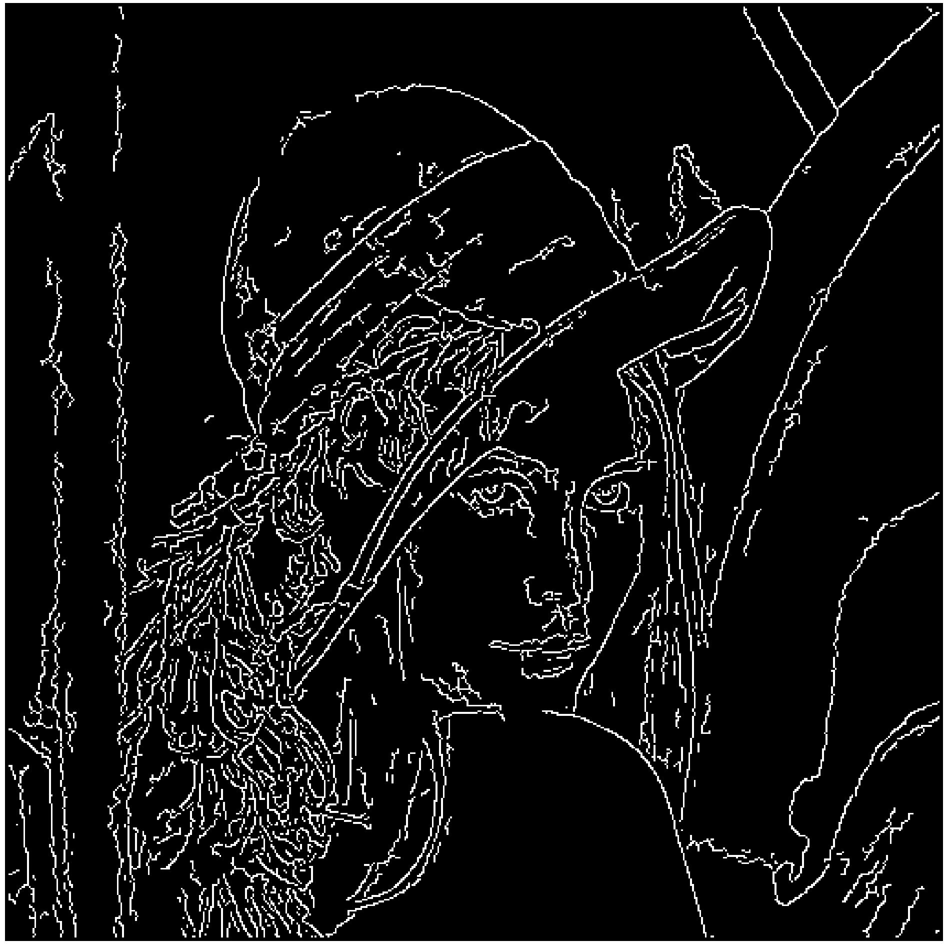


$\sigma = 3$, kernel 크기 17인 Gaussian filter

Canny edge detection

- **Example**

Noise image



Canny edge detection

Canny edge detection

1. **Smoothing image for noise reduction**
2. **Computing Image gradient to find edge candidates**

   $\Rightarrow$ DoG filtering
3. **Localizing edge: find peak of the first order derivative of image**

   $\Rightarrow$ Non-maximum suppression
4. **Thresholding edge**

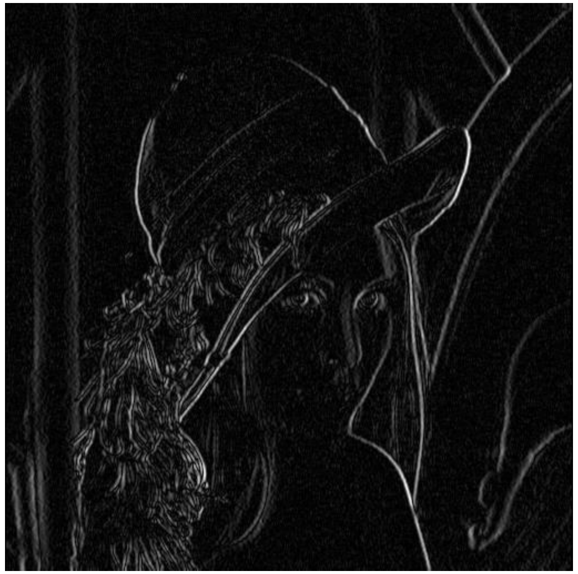
   $\Rightarrow$ Double thresholding

   $\Rightarrow$ Determine edge

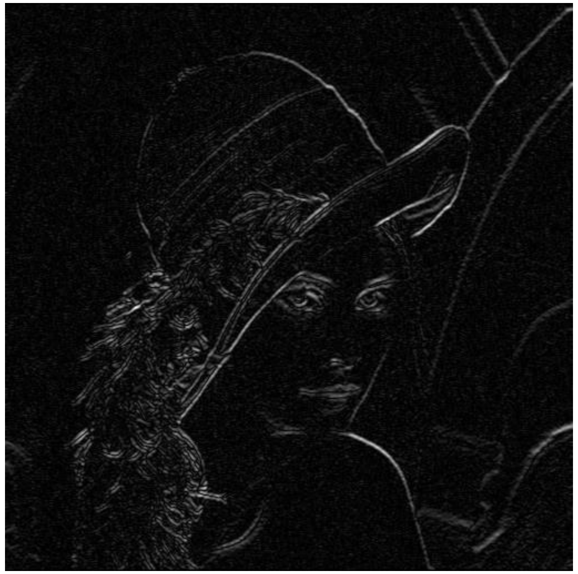
Canny edge detection

- **DoG filtering**
  - 5x5 DoG filter (Sigma 1)

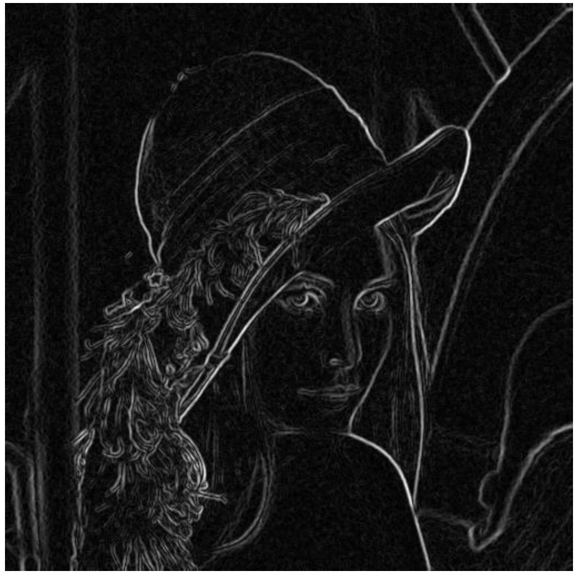
f



∇f_x



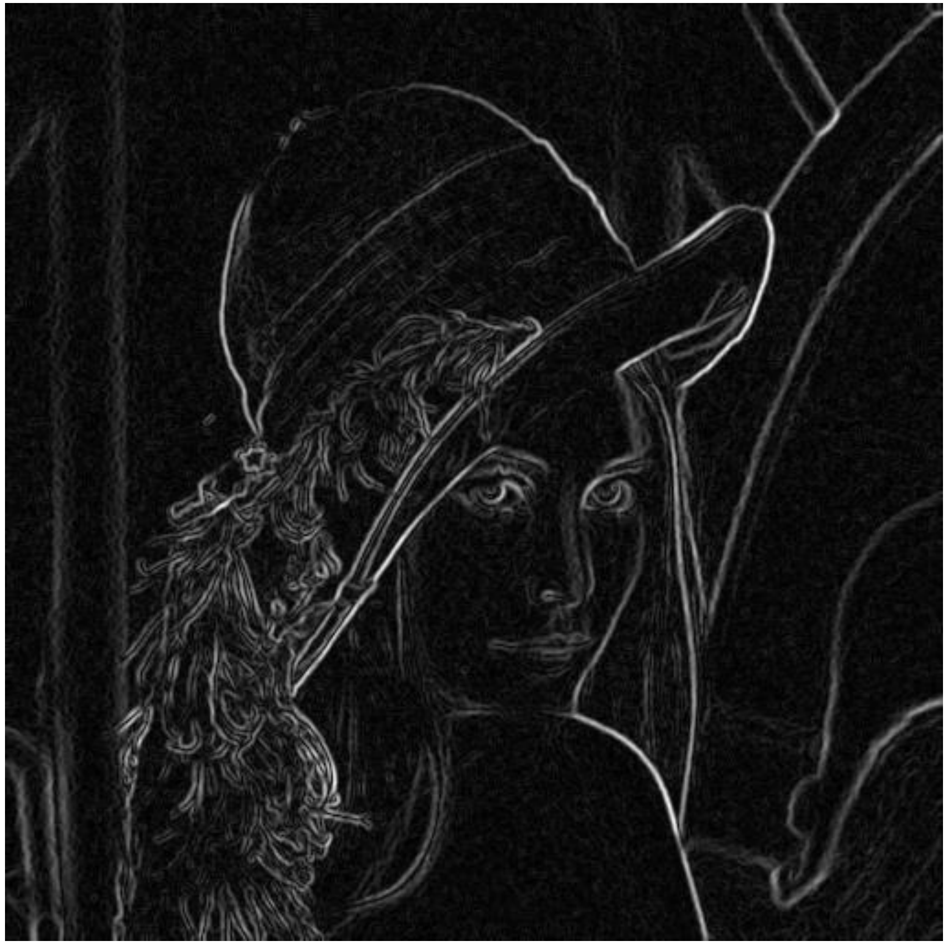
∇f_y



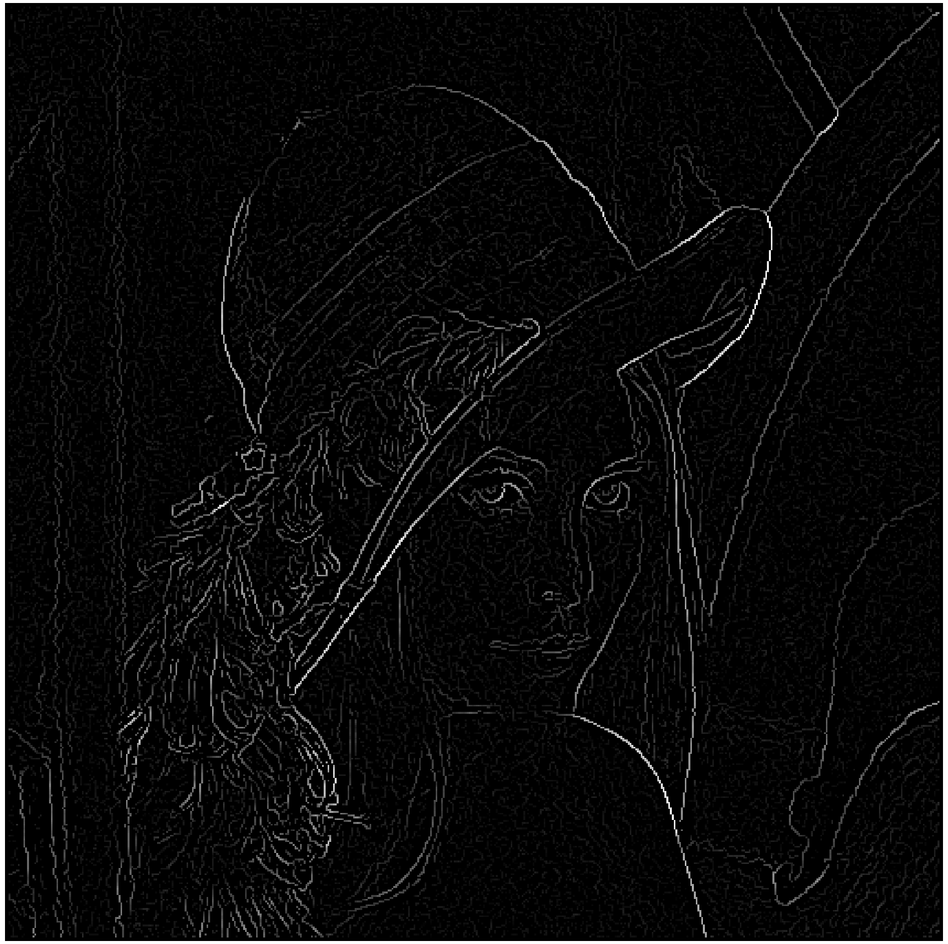
M

Canny edge detection

- **Non-maximum suppression**



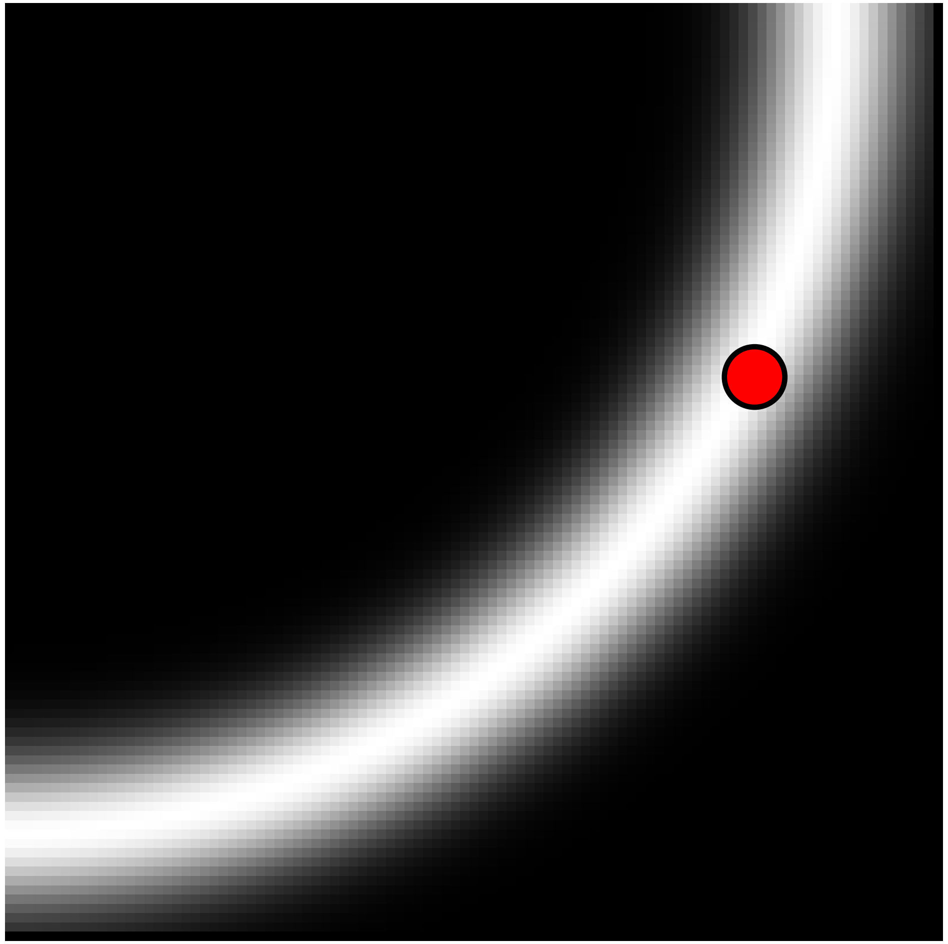
Magnitude



Non-maximum suppression

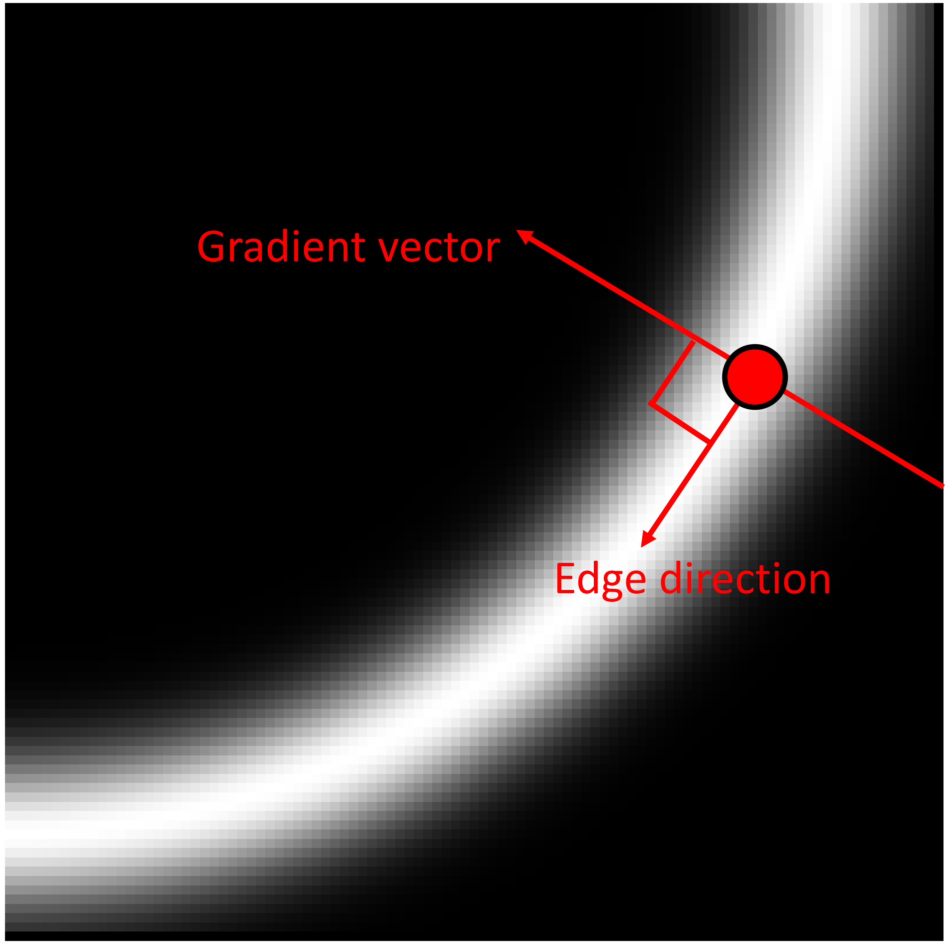
Canny edge detection

- **Non-maximum suppression**



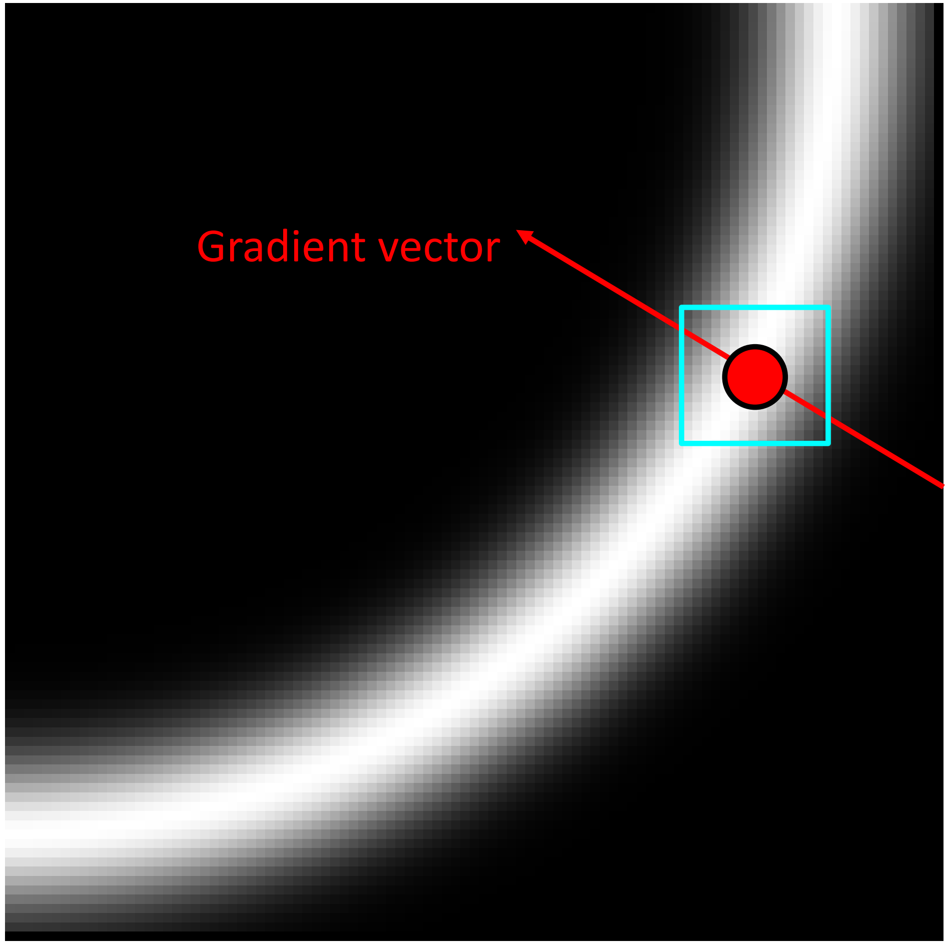
Canny edge detection

- **Non-maximum suppression**



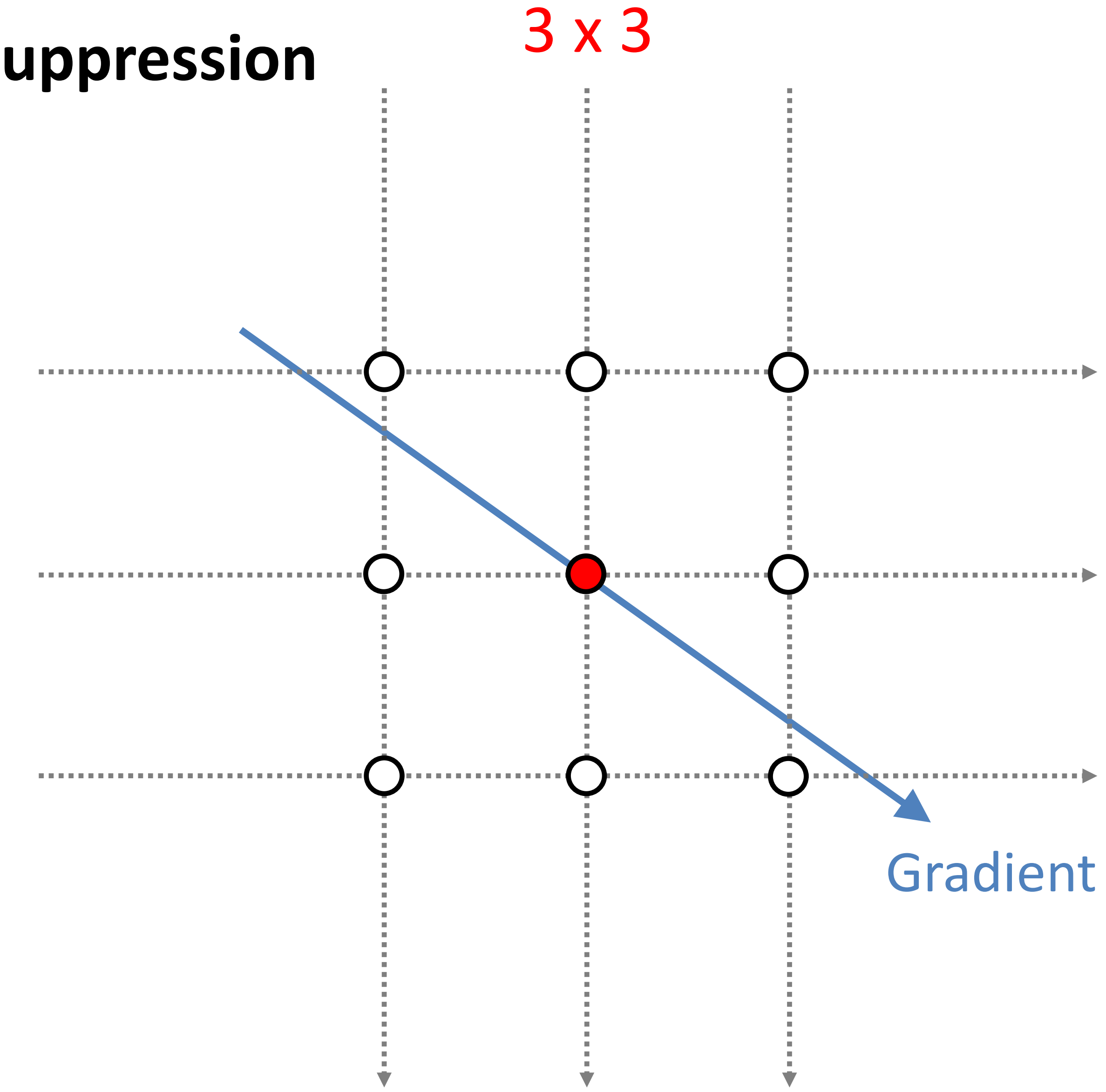
Canny edge detection

- **Non-maximum suppression**



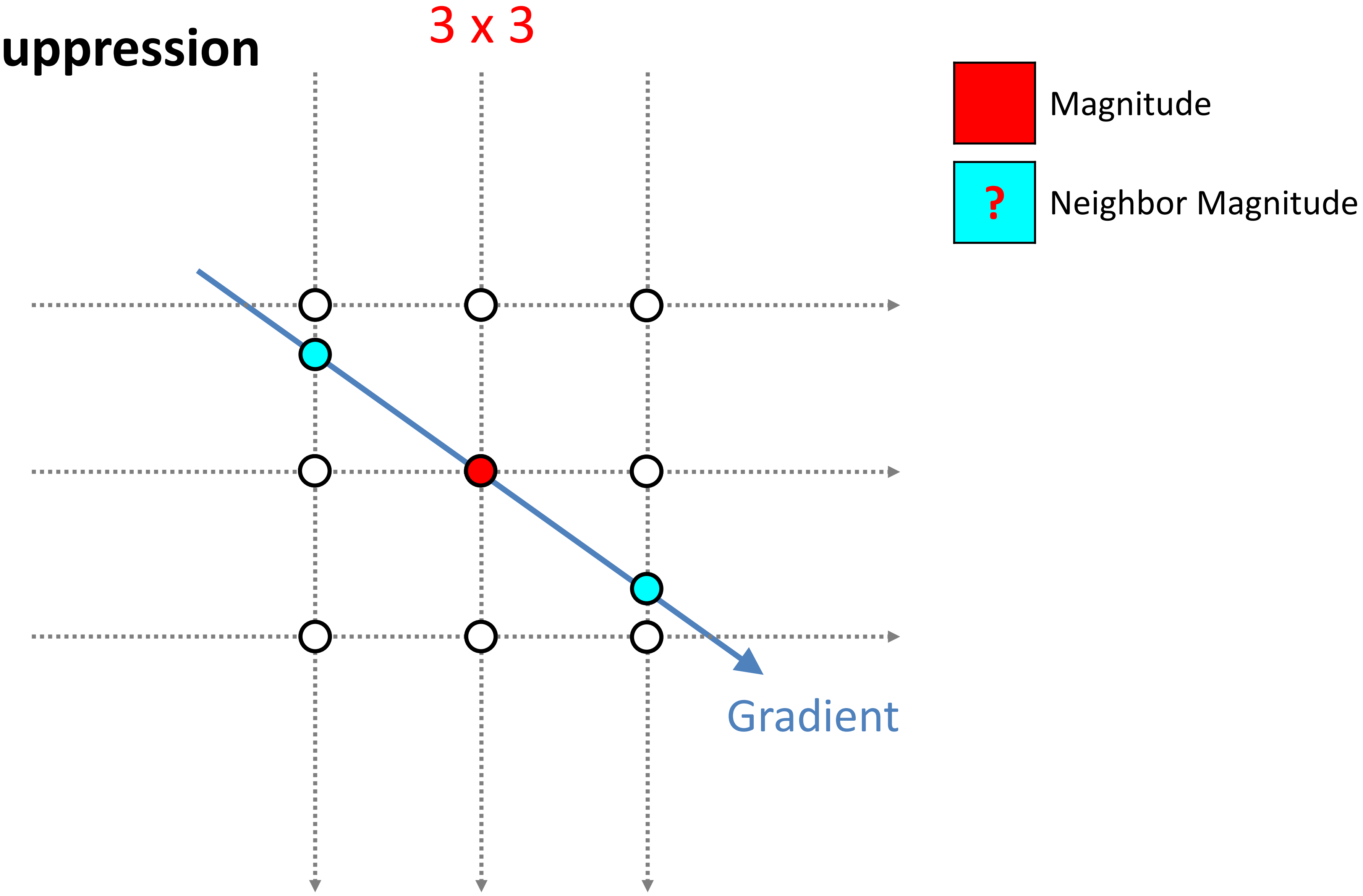
Canny edge detection

- **Non-maximum suppression**



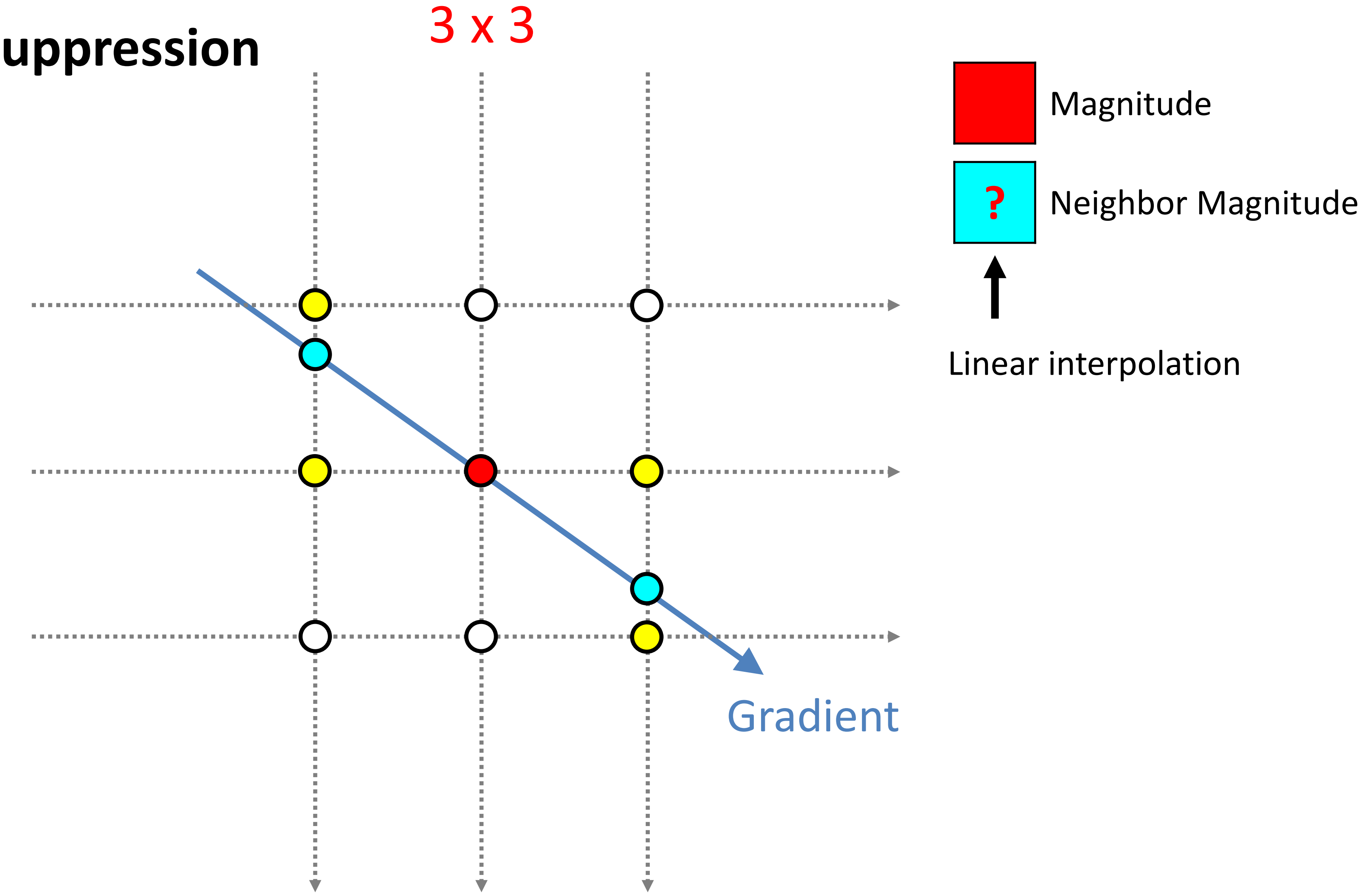
Canny edge detection

- **Non-maximum suppression**



Canny edge detection

- **Non-maximum suppression**



3 x 3

Gradient

Magnitude

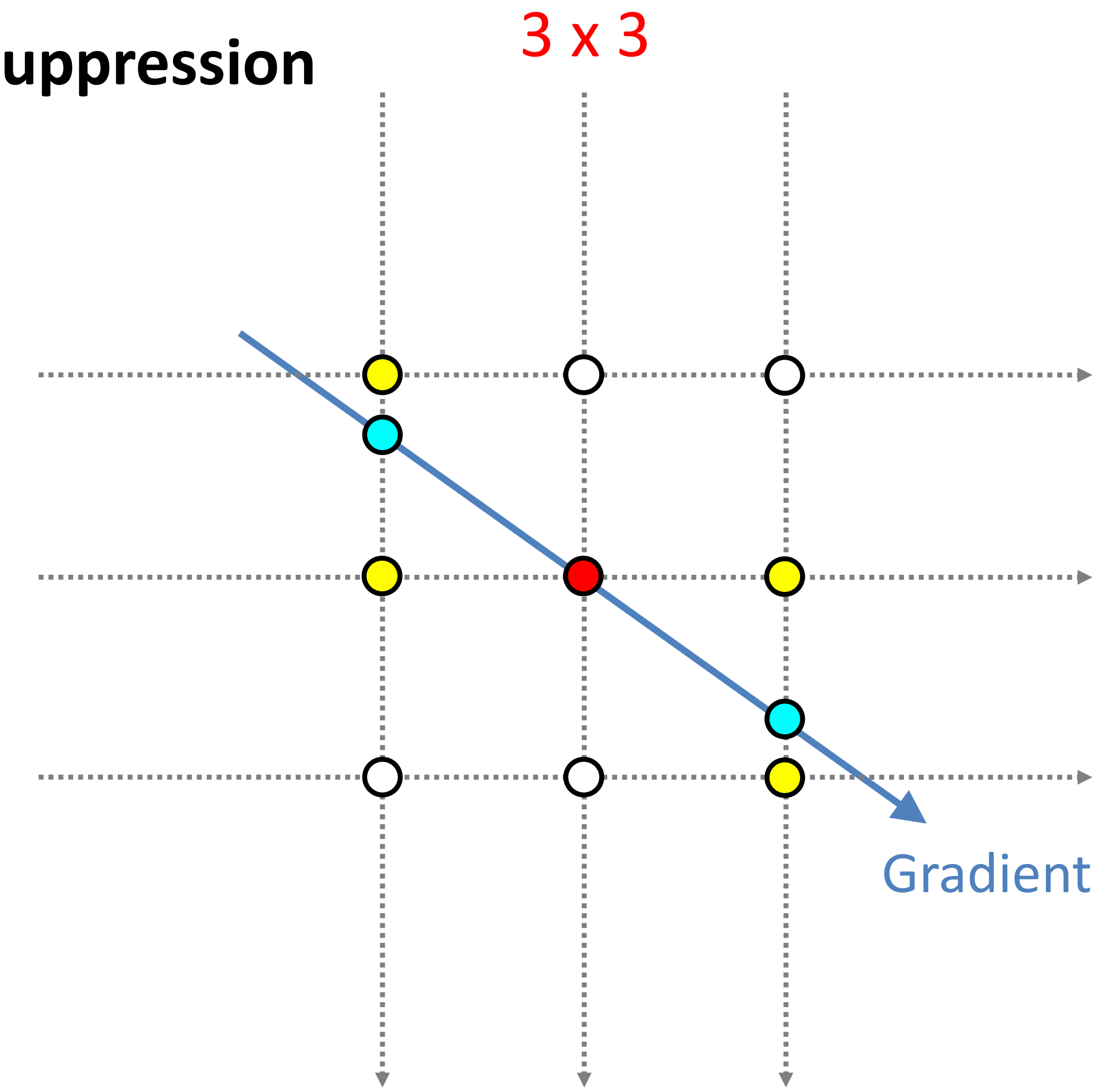
?

Neighbor Magnitude

Linear interpolation

Canny edge detection

- **Non-maximum suppression**



🟥 Magnitude

? Neighbor Magnitude

↑

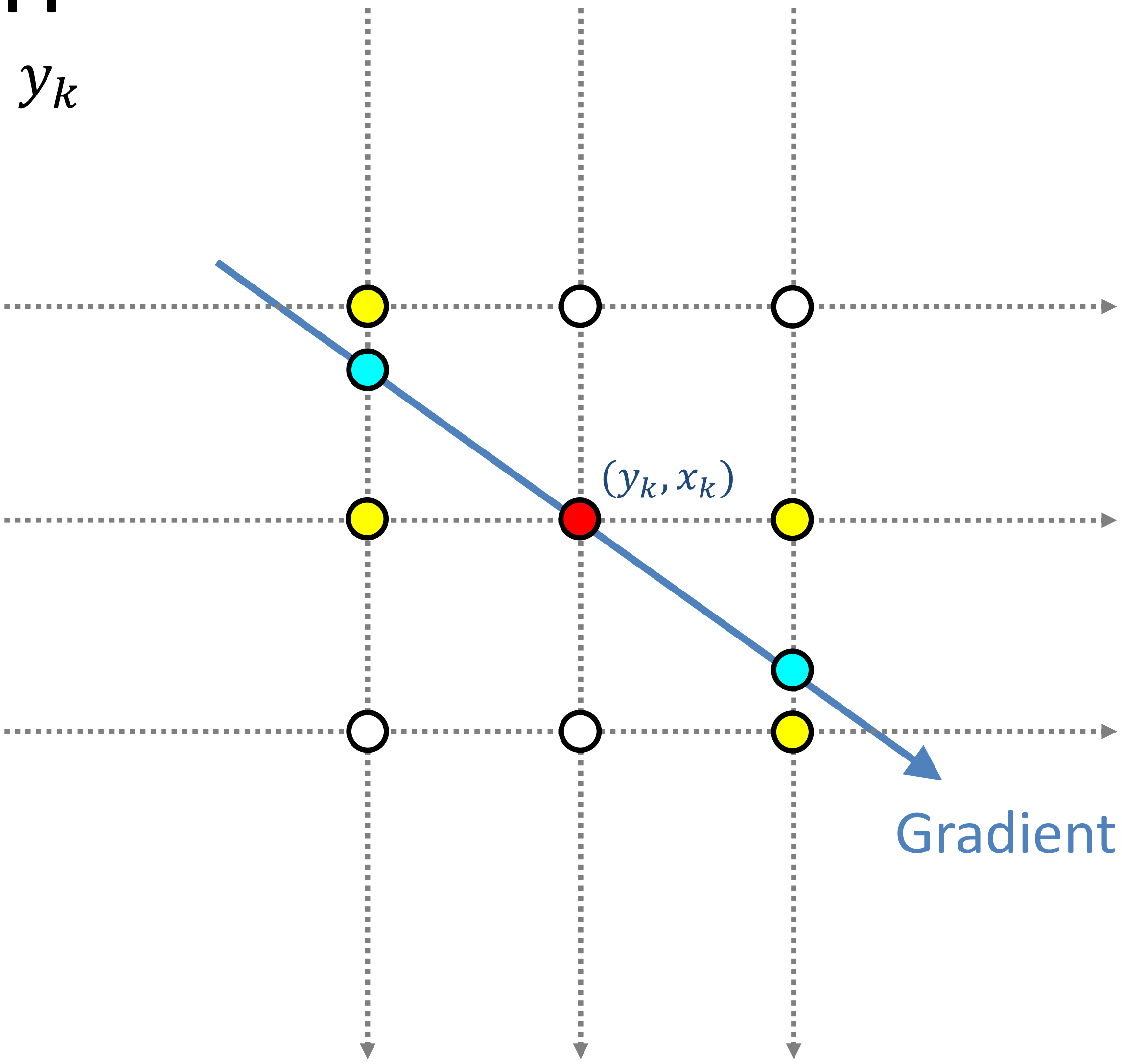
Linear interpolation

바닥함수 사용할 때:
int()말고 np.floor()사용
int(-1.5) -> -1
np.floor(-1.5) -> -2

Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$

3 x 3



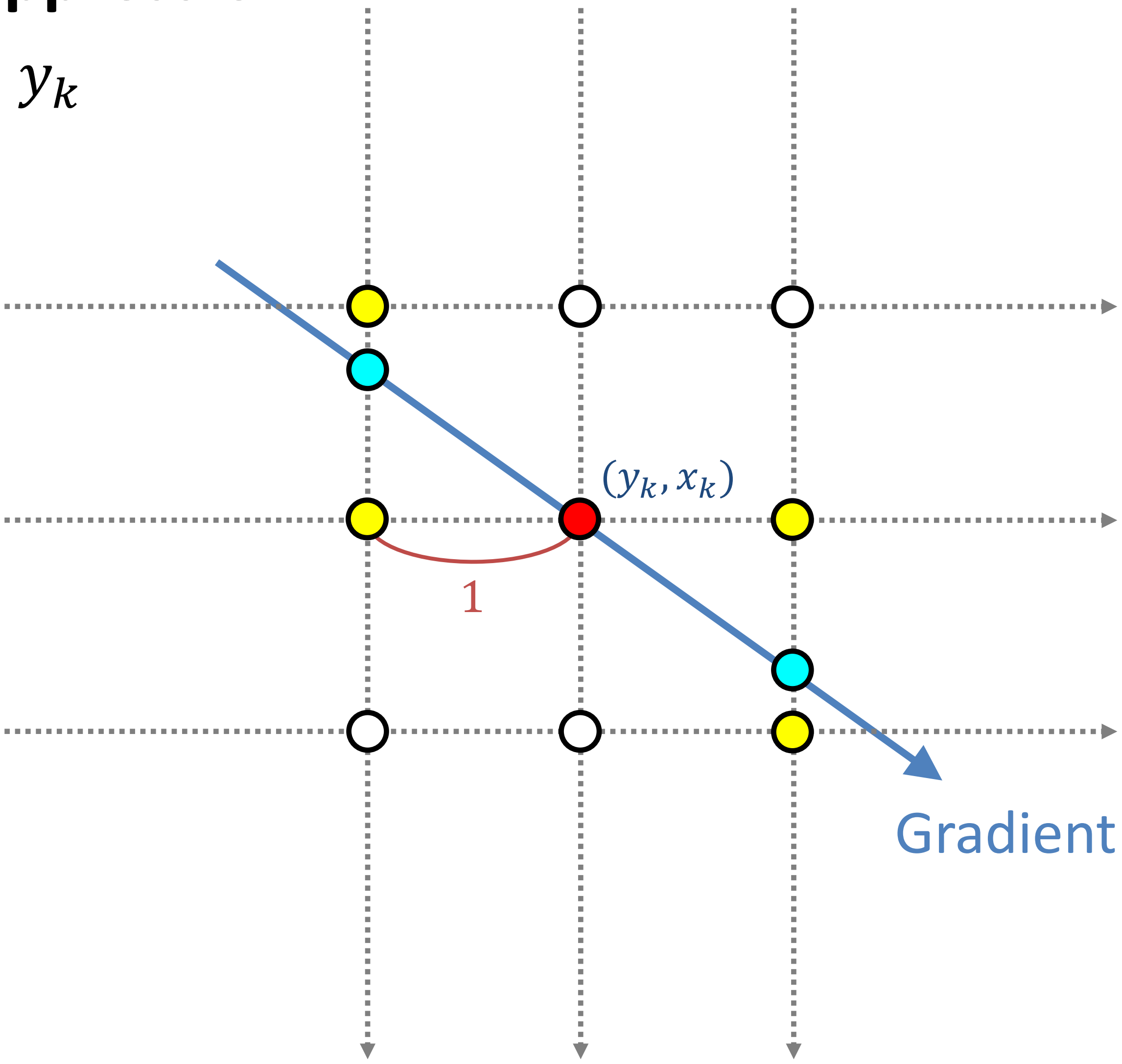
🟥 Magnitude

🟦 Neighbor Magnitude

Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$

3 x 3

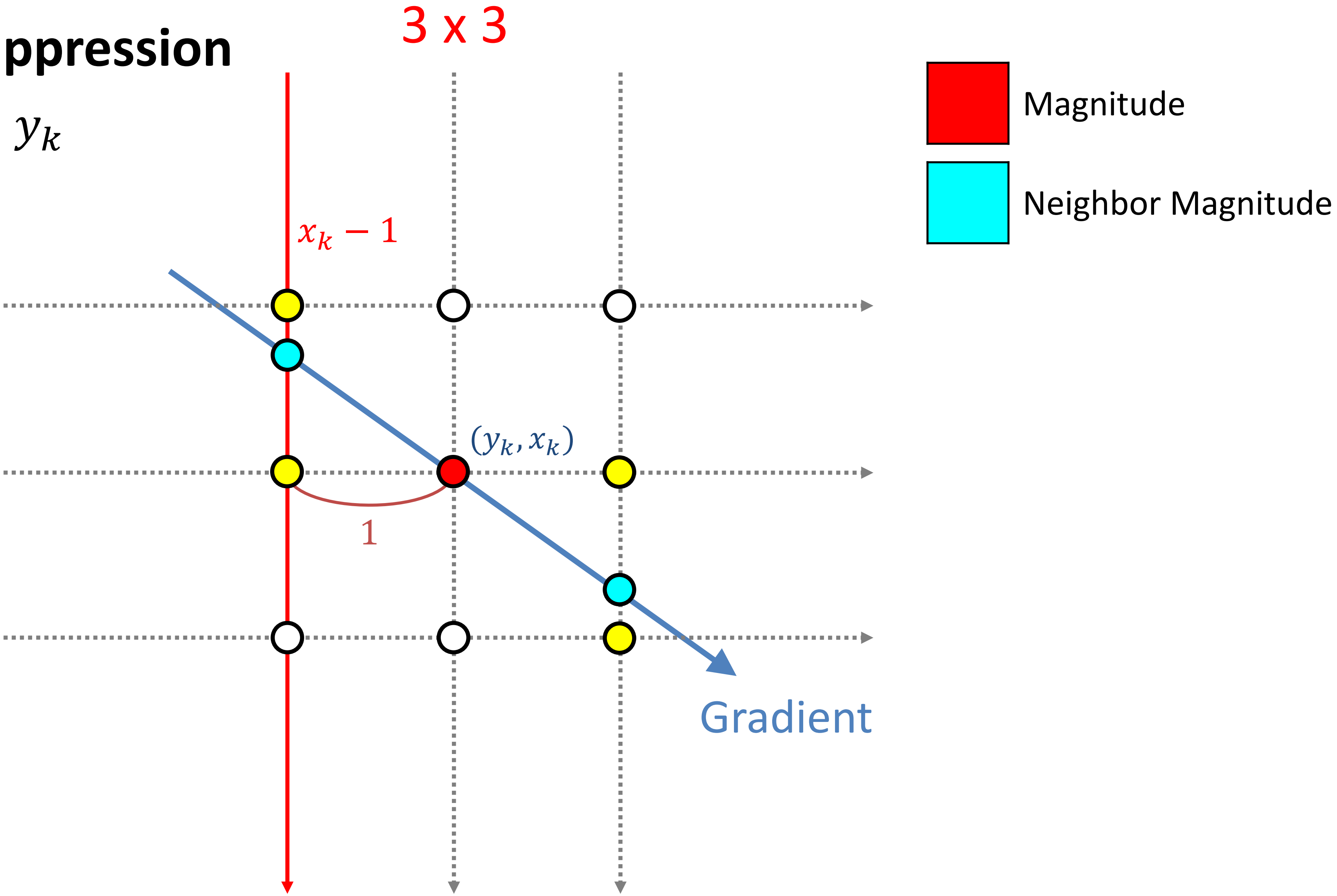


🟥 Magnitude

🟦 Neighbor Magnitude

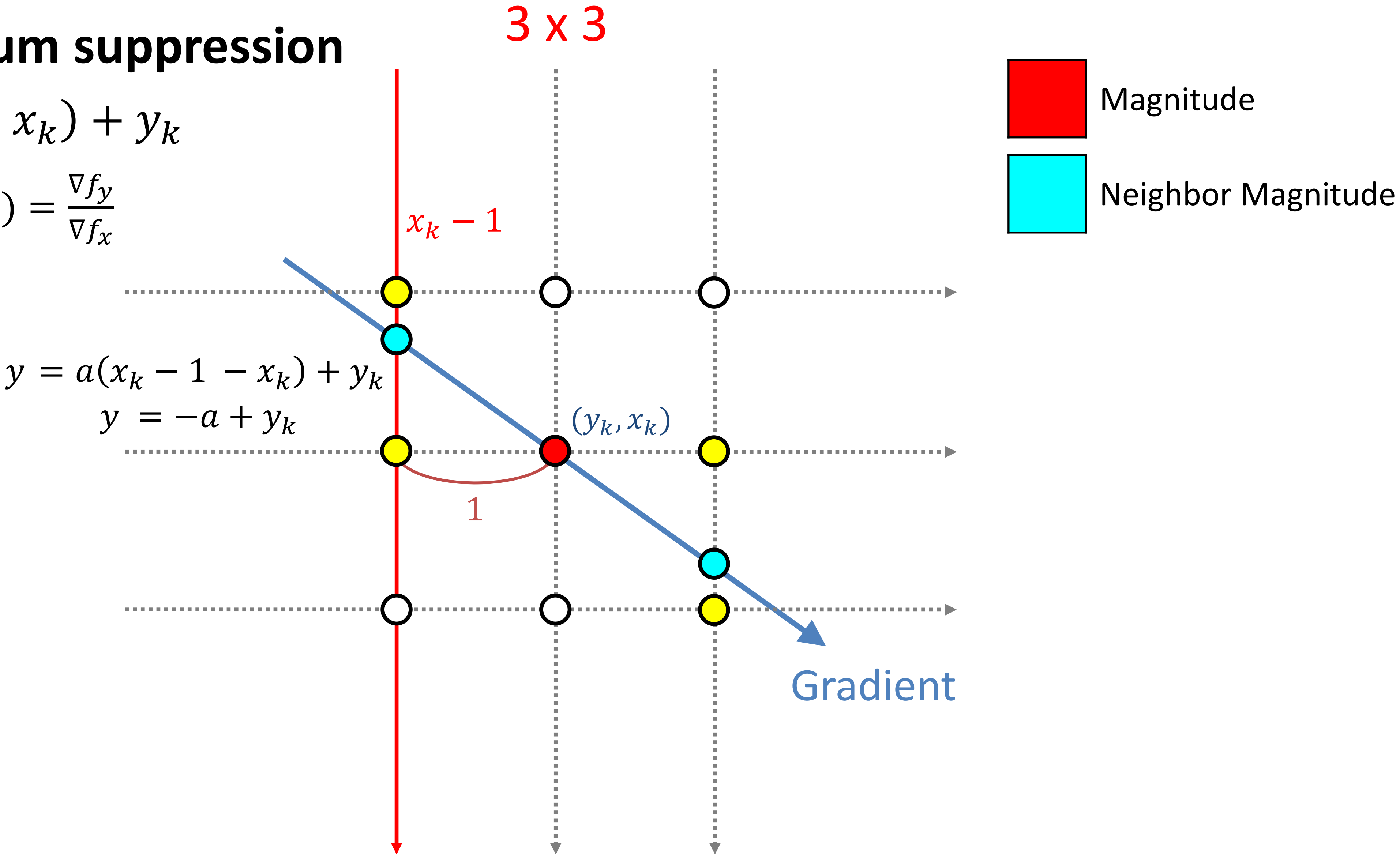
Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$



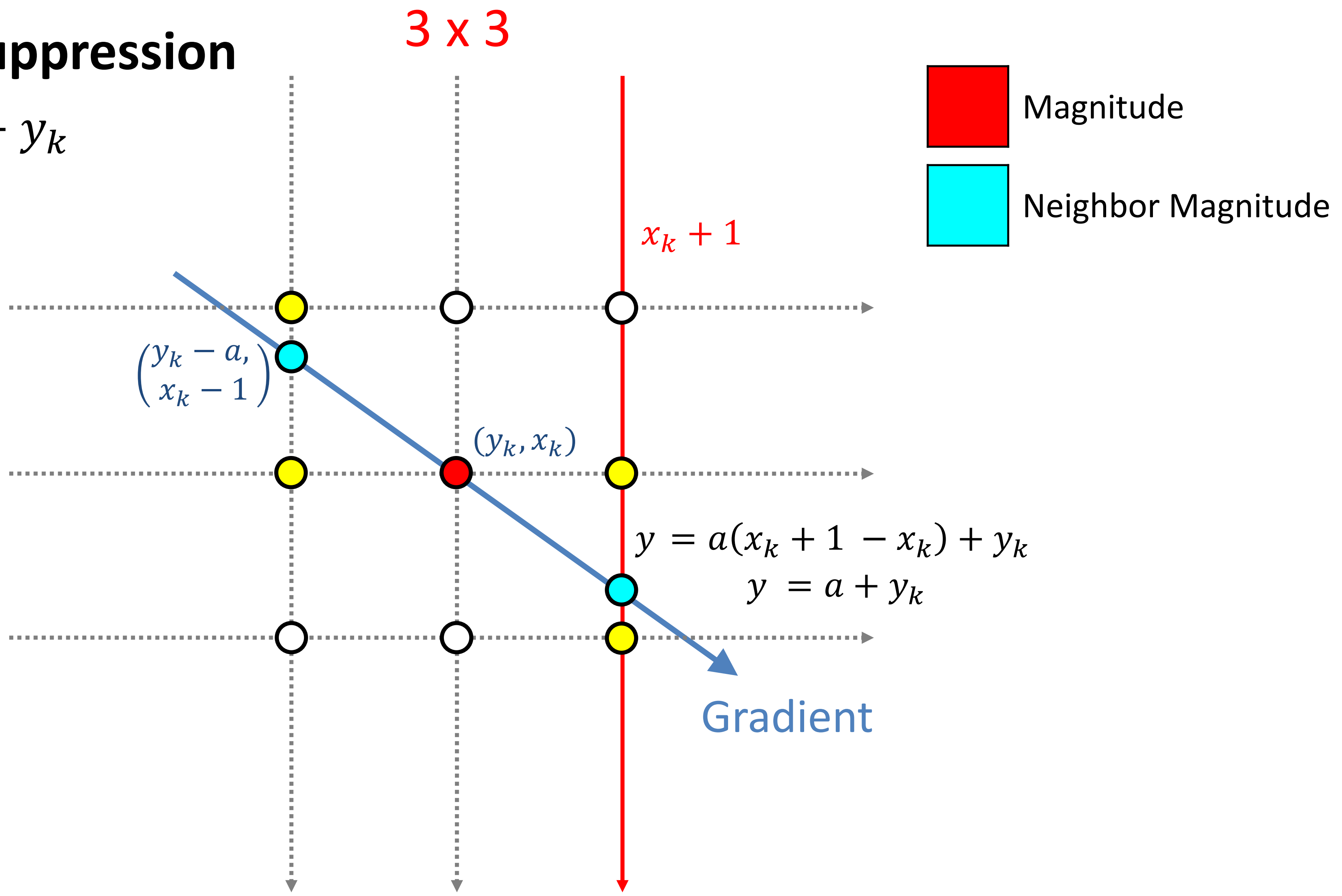
Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - a(기울기) $= \frac{\nabla f_y}{\nabla f_x}$



Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$

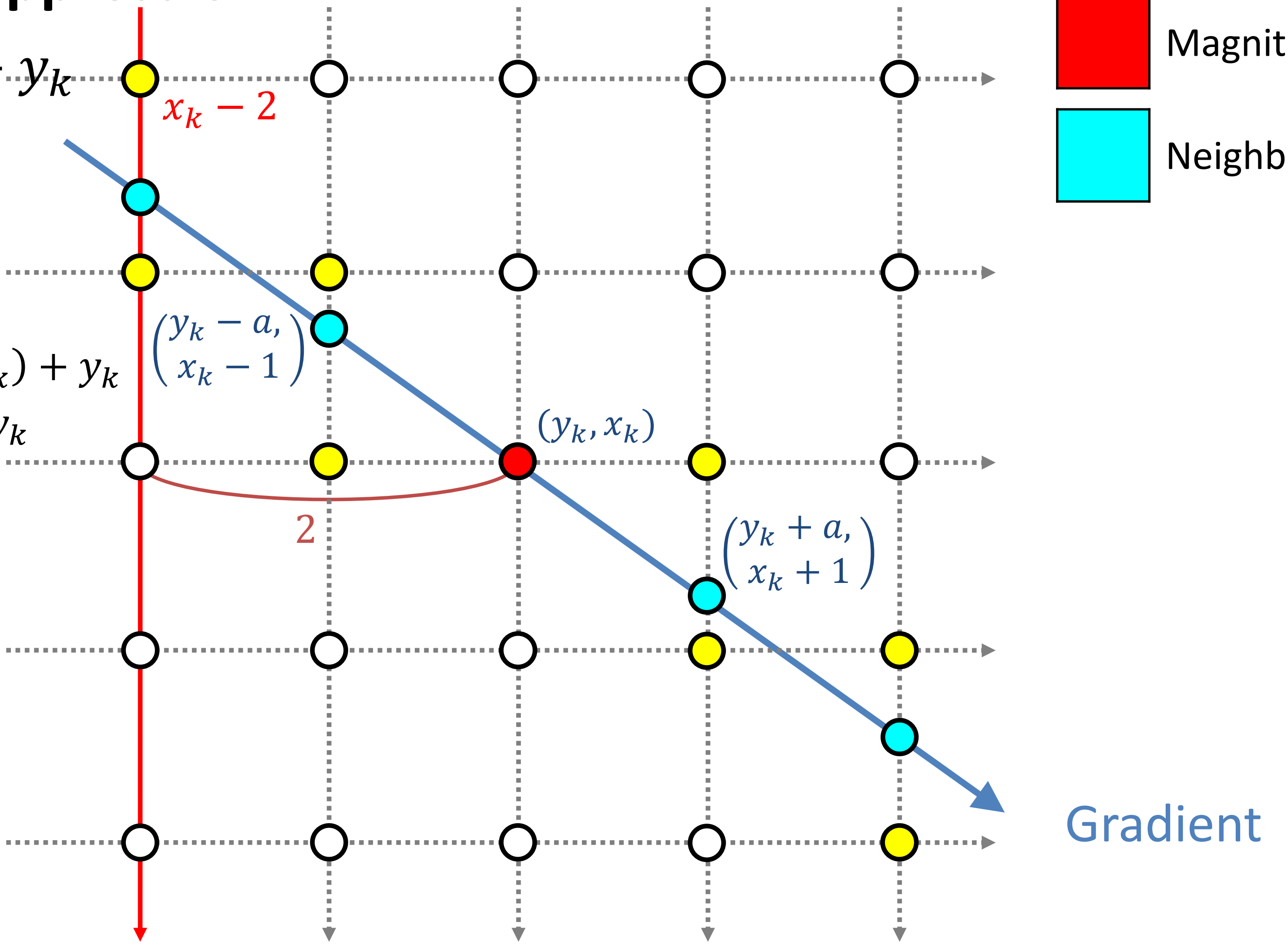

3 x 3
Magnitude
Neighbor Magnitude
$x_k + 1$
$\begin{pmatrix} y_k - a, \\ x_k - 1 \end{pmatrix}$
(y_k, x_k)
$y = a(x_k + 1 - x_k) + y_k$
$y = a + y_k$
Gradient

Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$

5 x 5

$$y = a(x_k - 2 - x_k) + y_k$$
$$y = -2a + y_k$$



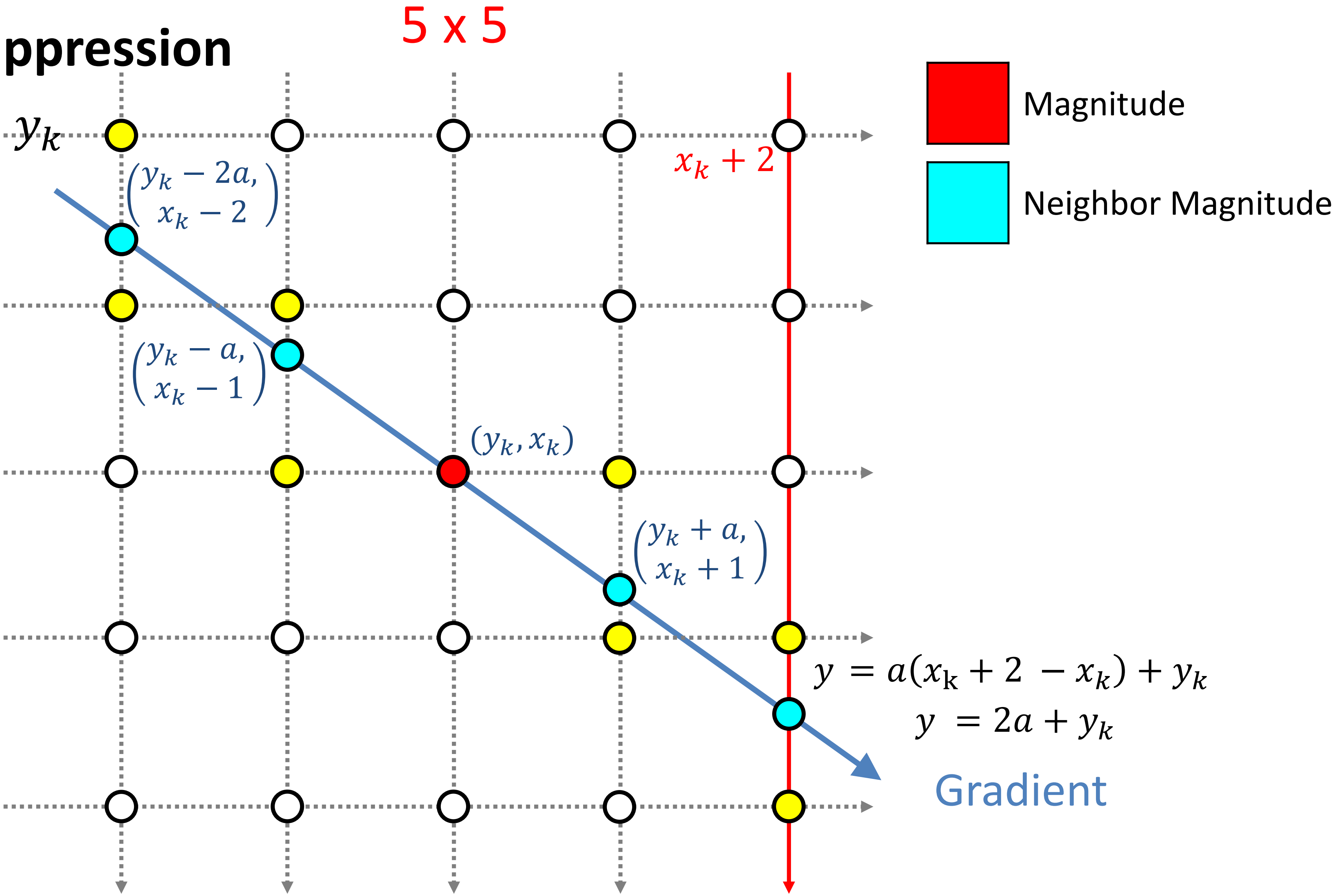
$x_k - 2$
$\begin{pmatrix} y_k - a, \\ x_k - 1 \end{pmatrix}$
(y_k, x_k)
2
$\begin{pmatrix} y_k + a, \\ x_k + 1 \end{pmatrix}$
Gradient

🟥 Magnitude

🟦 Neighbor Magnitude

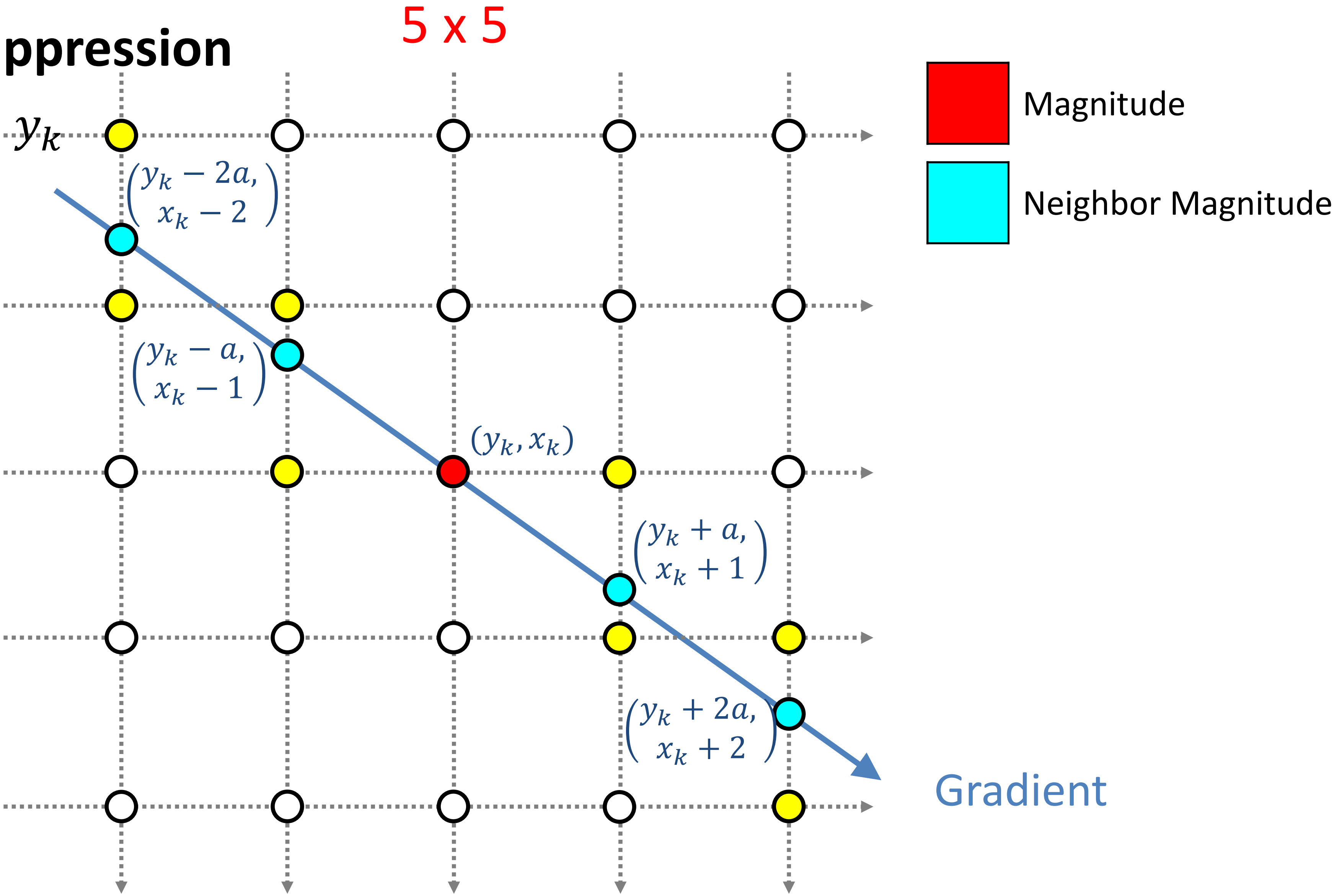
Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - a(기울기) $= \frac{\nabla f_y}{\nabla f_x}$


5 x 5
🟥 Magnitude
🟦 Neighbor Magnitude
$x_k + 2$
$\begin{pmatrix} y_k - 2a, \\ x_k - 2 \end{pmatrix}$
$\begin{pmatrix} y_k - a, \\ x_k - 1 \end{pmatrix}$
(y_k, x_k)
$\begin{pmatrix} y_k + a, \\ x_k + 1 \end{pmatrix}$
$y = a(x_k + 2 - x_k) + y_k$
$y = 2a + y_k$
Gradient

Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$


5 x 5
$\begin{pmatrix} y_k - 2a, \\ x_k - 2 \end{pmatrix}$
$\begin{pmatrix} y_k - a, \\ x_k - 1 \end{pmatrix}$
(y_k, x_k)
$\begin{pmatrix} y_k + a, \\ x_k + 1 \end{pmatrix}$
$\begin{pmatrix} y_k + 2a, \\ x_k + 2 \end{pmatrix}$
🟥 Magnitude
🟦 Neighbor Magnitude
Gradient

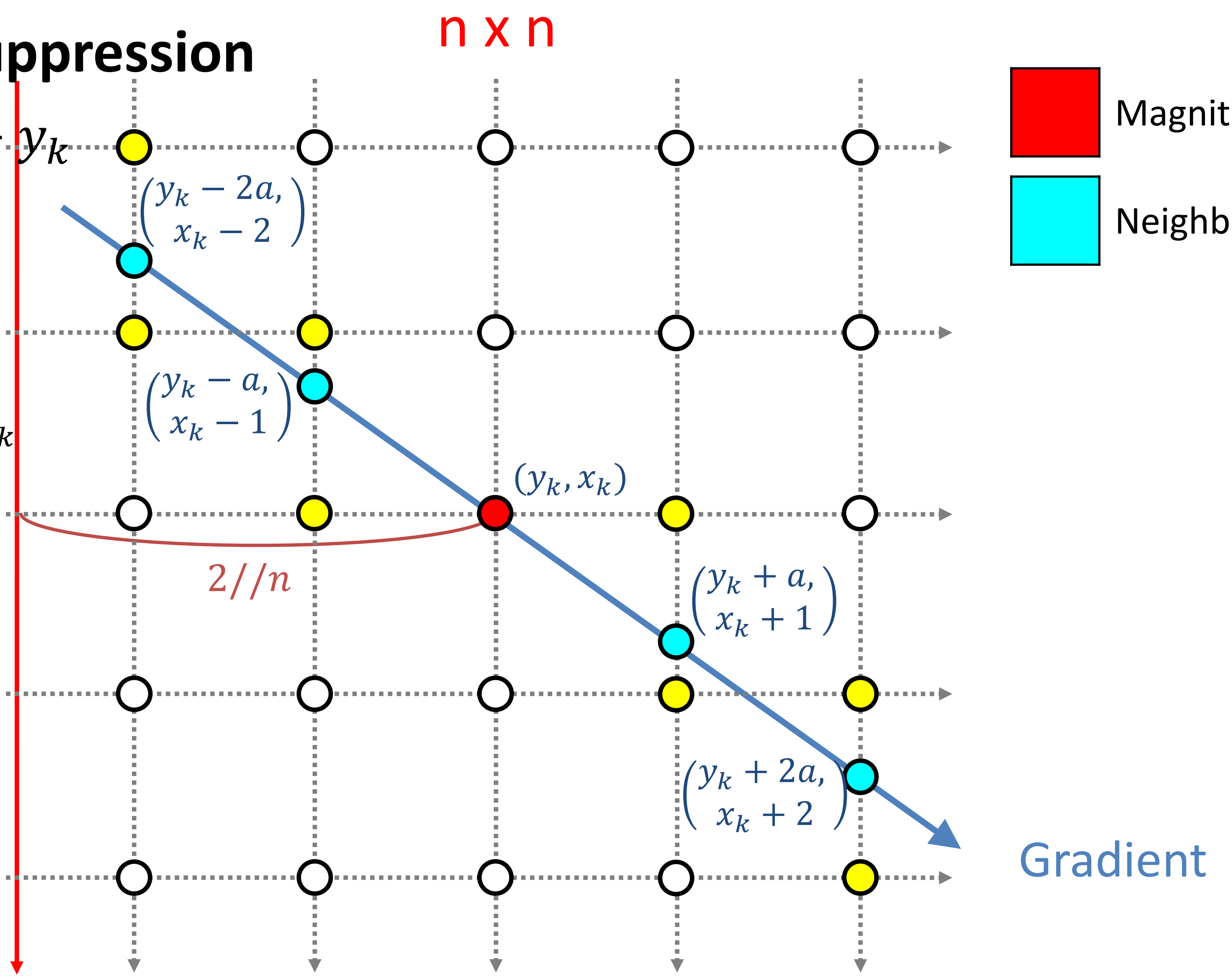
Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$

$x_k - (2//n)$

$y = a \times (x_k - (2//n) - x_k) + y_k$

$y = -a \times (2//n) + y_k$

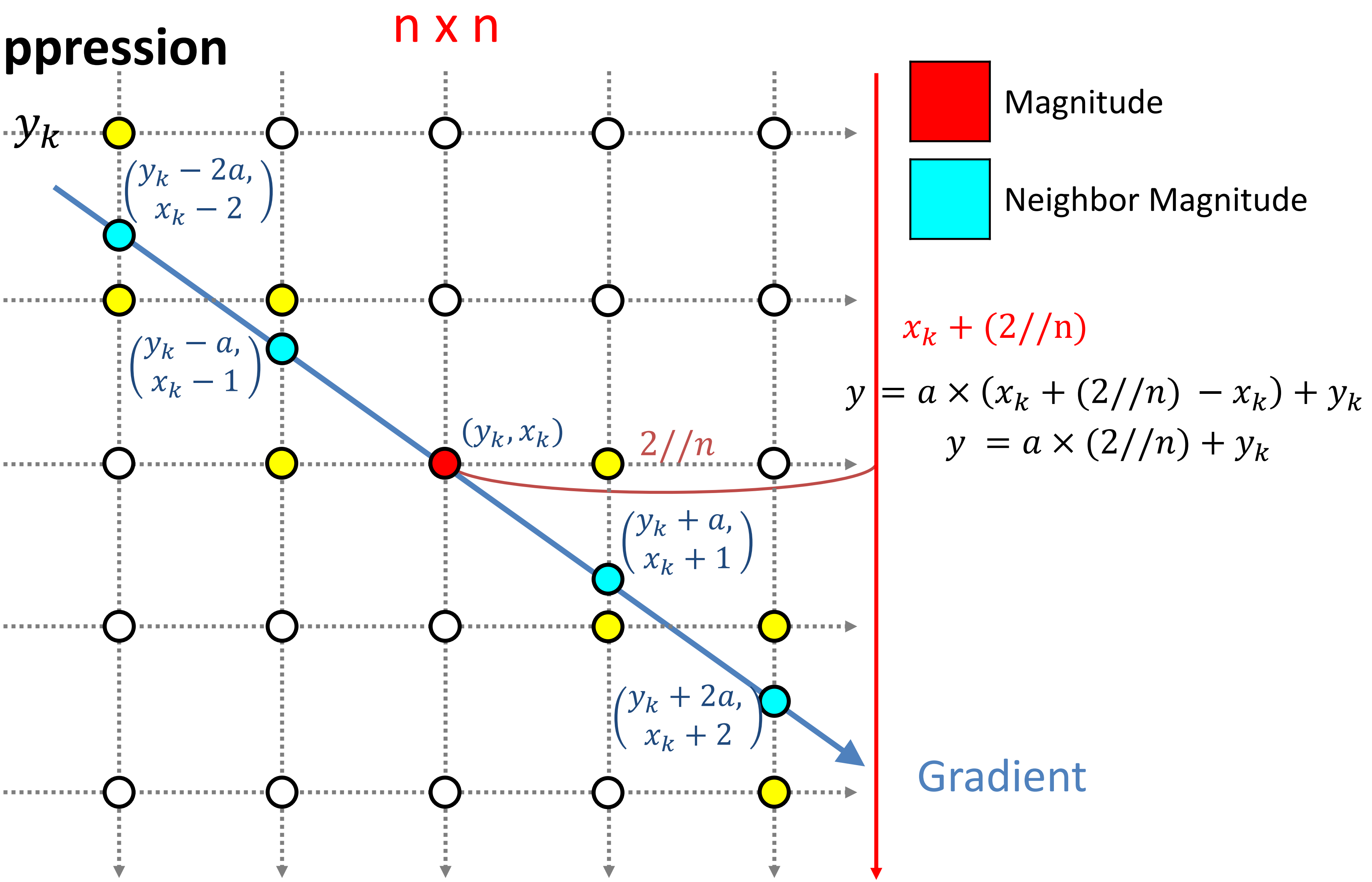

n x n
$\begin{pmatrix} y_k - 2a, \\ x_k - 2 \end{pmatrix}$
$\begin{pmatrix} y_k - a, \\ x_k - 1 \end{pmatrix}$
(y_k, x_k)
$2//n$
$\begin{pmatrix} y_k + a, \\ x_k + 1 \end{pmatrix}$
$\begin{pmatrix} y_k + 2a, \\ x_k + 2 \end{pmatrix}$
Gradient

🟥 Magnitude

🟦 Neighbor Magnitude

Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$

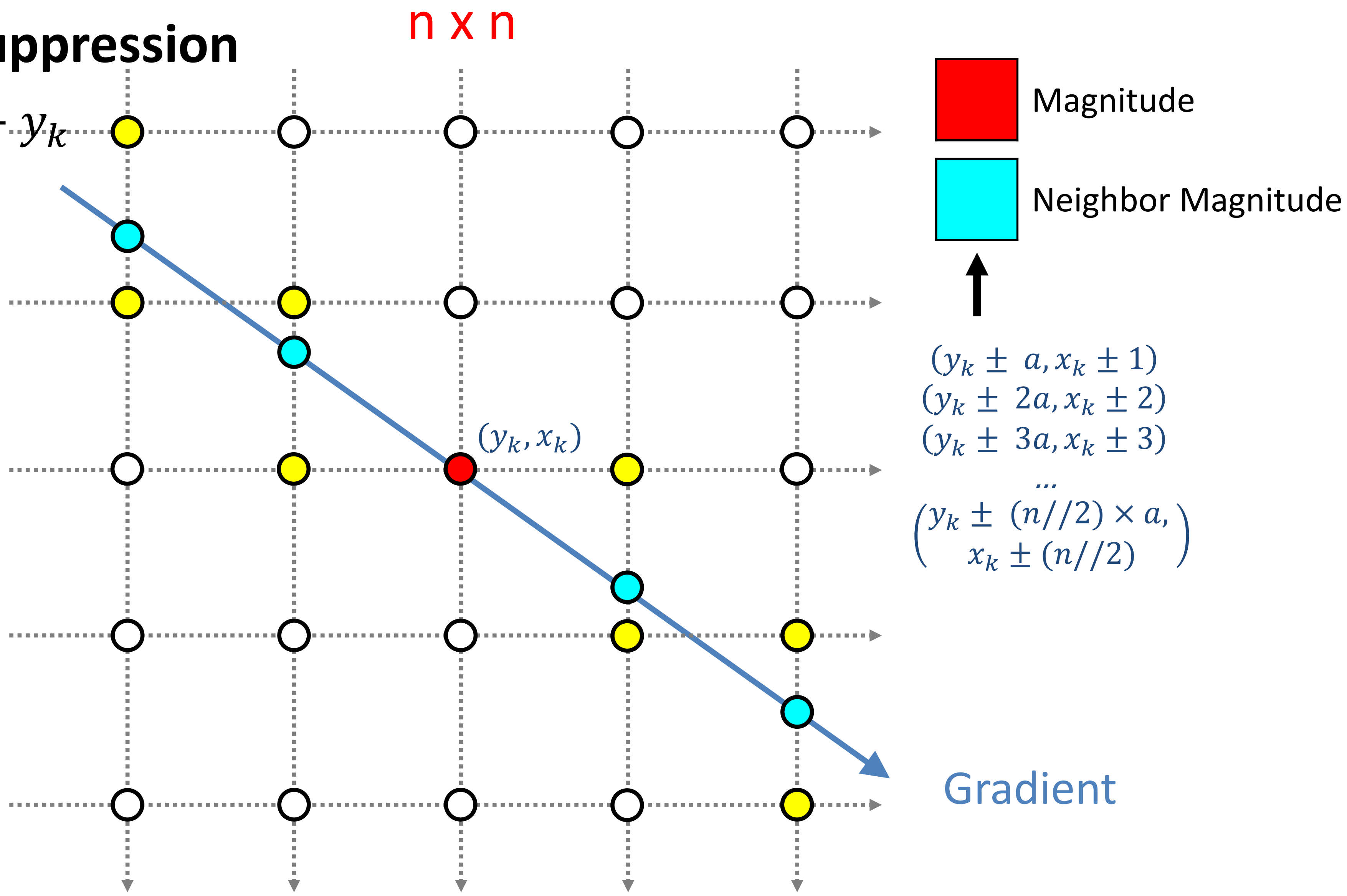

n x n
🟥 Magnitude
🟦 Neighbor Magnitude
$\begin{pmatrix} y_k - 2a, \\ x_k - 2 \end{pmatrix}$
$\begin{pmatrix} y_k - a, \\ x_k - 1 \end{pmatrix}$
(y_k, x_k)
$2//n$
$\begin{pmatrix} y_k + a, \\ x_k + 1 \end{pmatrix}$
$\begin{pmatrix} y_k + 2a, \\ x_k + 2 \end{pmatrix}$
$x_k + (2//\mathrm{n})$
Gradient

$$y = a \times (x_k + (2//n) - x_k) + y_k$$

$$y = a \times (2//n) + y_k$$

Canny edge detection

- **Non-maximum suppression**
  - $- y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$



$(y_k \pm a, x_k \pm 1)$
$(y_k \pm 2a, x_k \pm 2)$
$(y_k \pm 3a, x_k \pm 3)$
…
$\begin{pmatrix} y_k \pm (n//2) \times a, \\ x_k \pm (n//2) \end{pmatrix}$

Canny edge detection

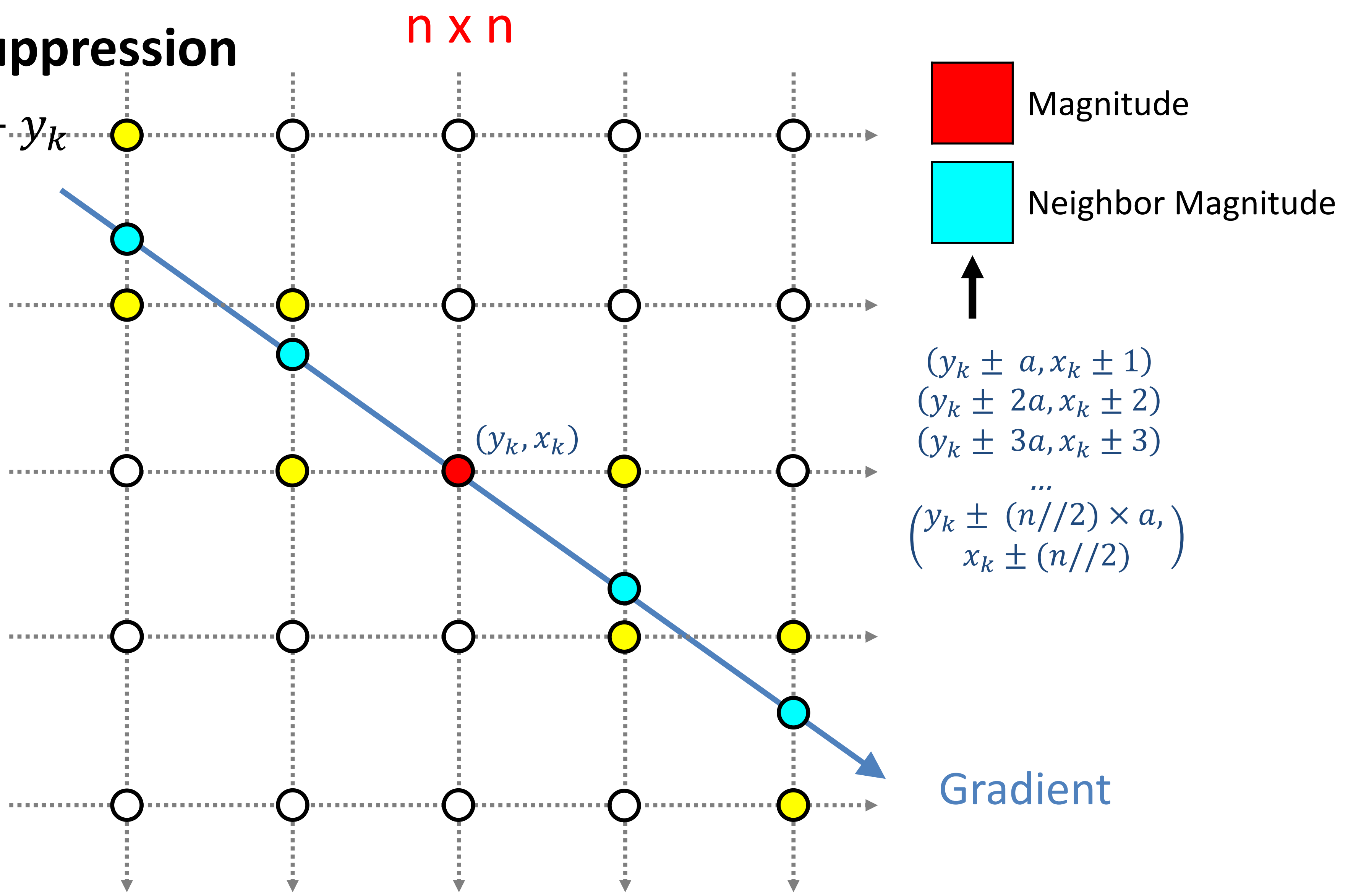
- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$

$$\downarrow$$

$$|\nabla f_y| < |\nabla f_x|$$

$$\left|\frac{\nabla f_y}{\nabla f_x}\right| < 1$$

$$|a| < 1$$



$$\uparrow$$

$(y_k \pm a, x_k \pm 1)$
$(y_k \pm 2a, x_k \pm 2)$
$(y_k \pm 3a, x_k \pm 3)$
...
$\begin{pmatrix} y_k \pm (n//2) \times a, \\ x_k \pm (n//2) \end{pmatrix}$

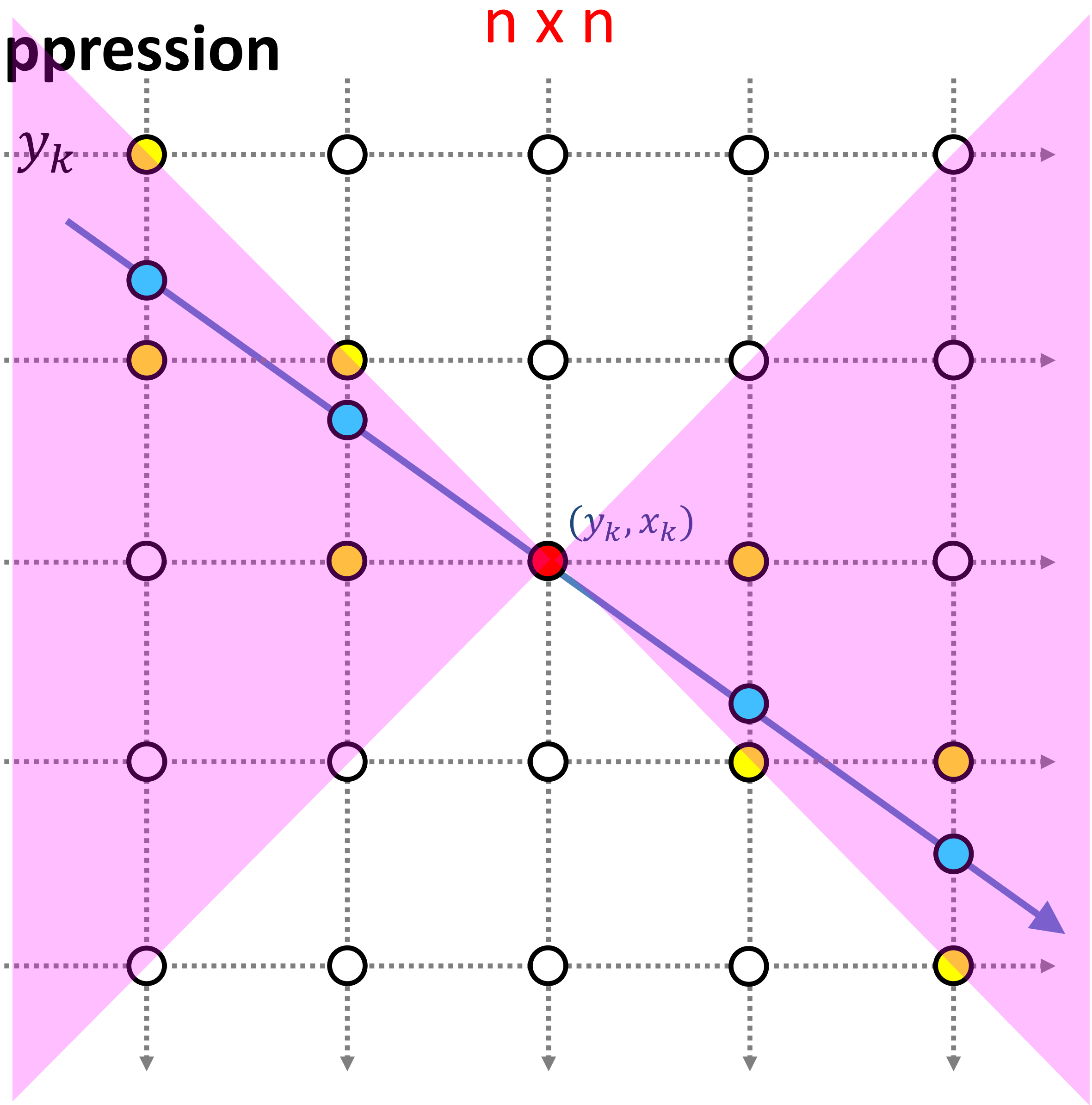
Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - a(기울기) $= \frac{\nabla f_y}{\nabla f_x}$

$$|\nabla f_y| < |\nabla f_x|$$

$$\left|\frac{\nabla f_y}{\nabla f_x}\right| < 1$$

$$|a| < 1$$



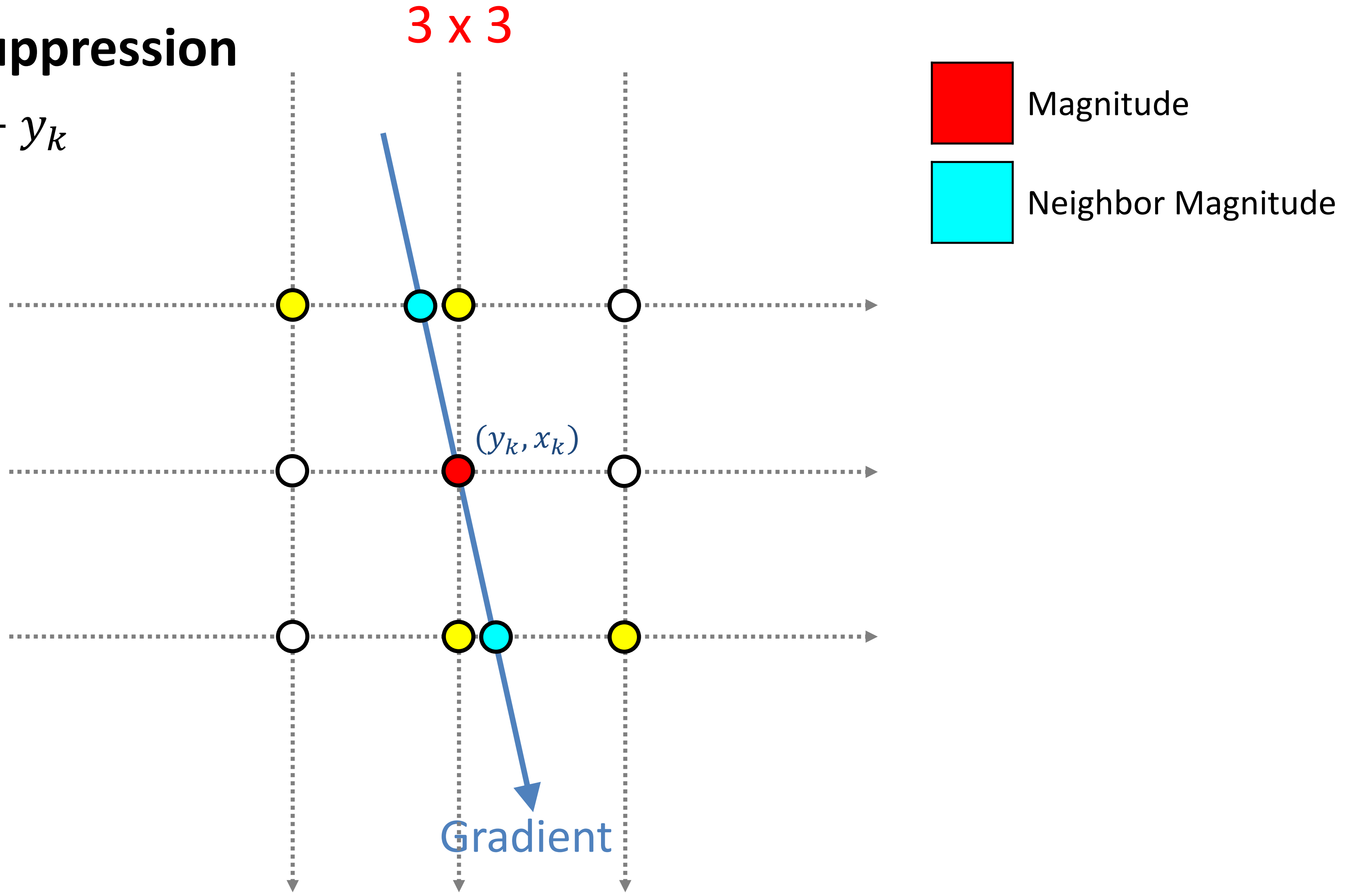
🟥 Magnitude

🟦 Neighbor Magnitude

$(y_k \pm a, x_k \pm 1)$
$(y_k \pm 2a, x_k \pm 2)$
$(y_k \pm 3a, x_k \pm 3)$
…
$\left(\begin{array}{c} y_k \pm (n//2) \times a, \\ x_k \pm (n//2) \end{array}\right)$

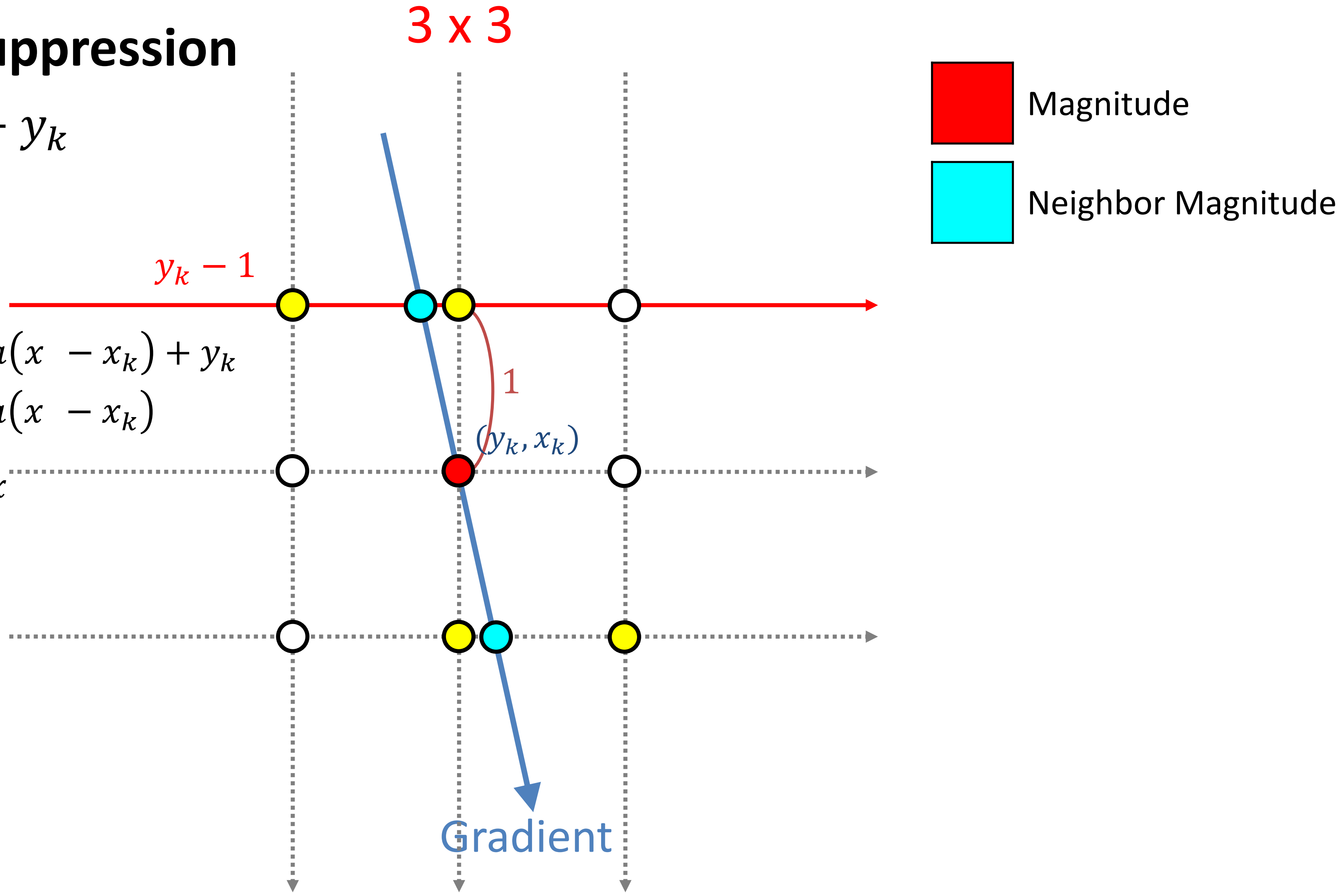
Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$



Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$



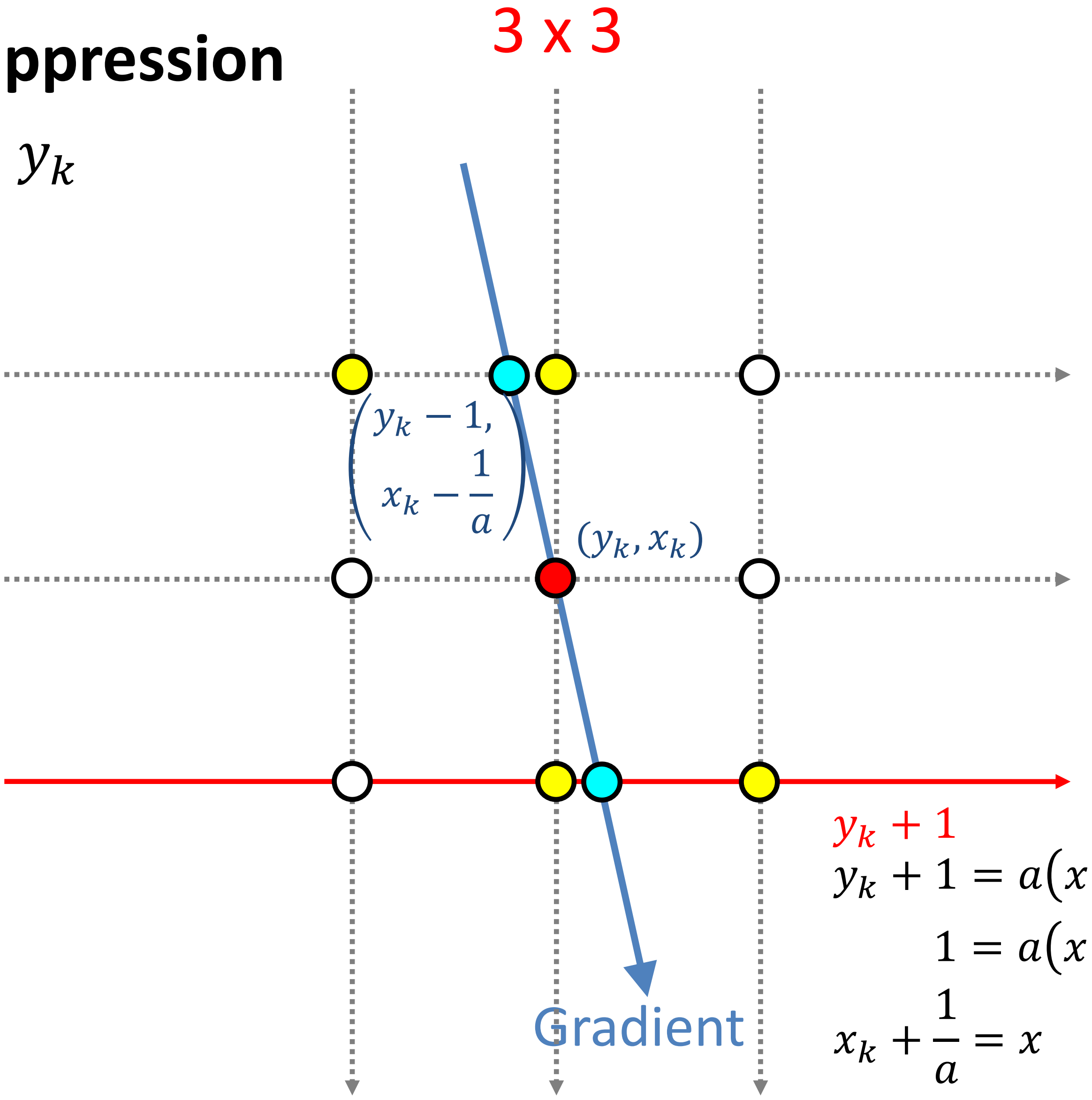
$$y_k - 1 = a(x - x_k) + y_k$$

$$-1 = a(x - x_k)$$

$$x_k - \frac{1}{a} = x$$

Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$



🟥 Magnitude

🟦 Neighbor Magnitude

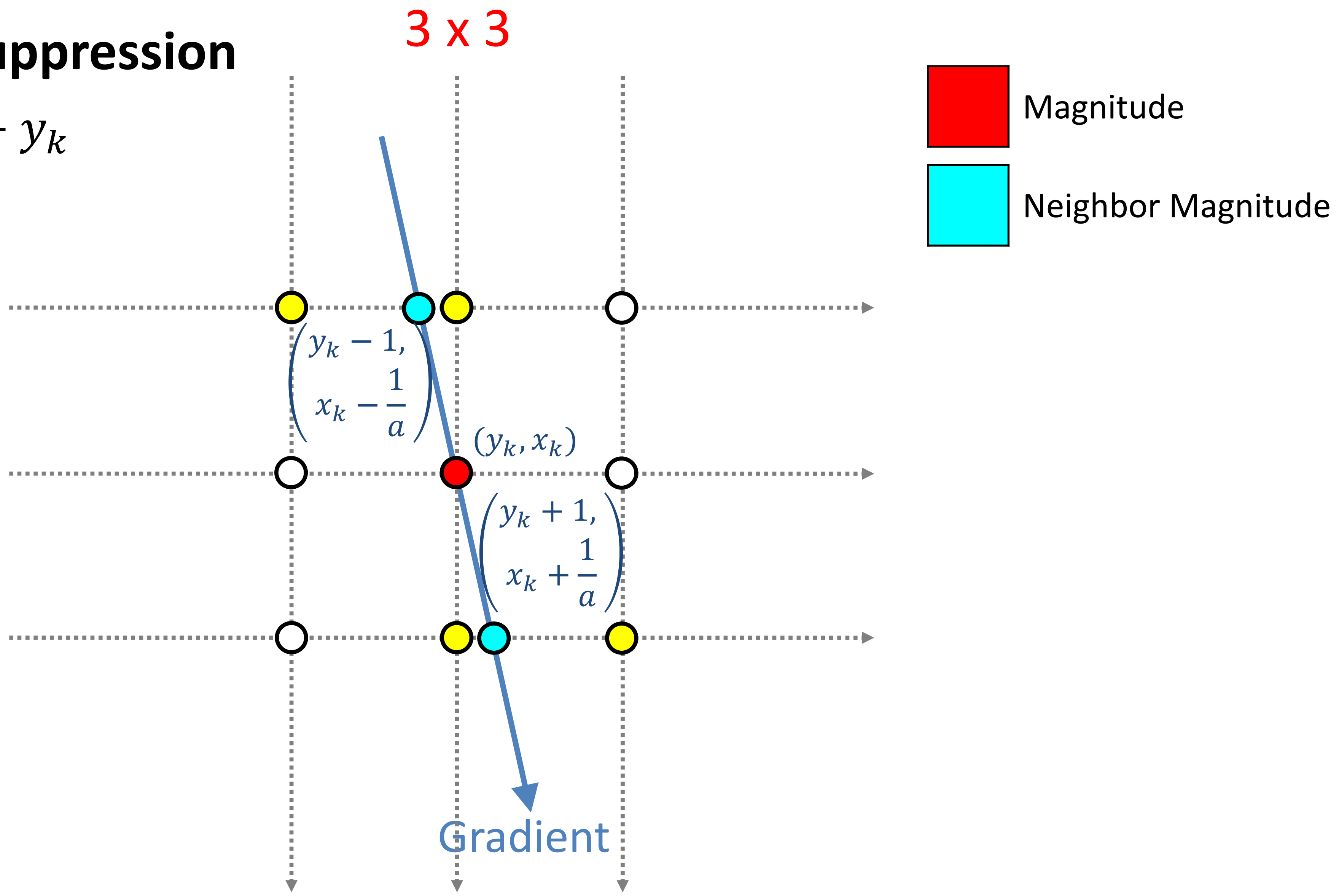
$$y_k + 1 = a(x - x_k) + y_k$$

$$1 = a(x - x_k)$$

$$x_k + \frac{1}{a} = x$$

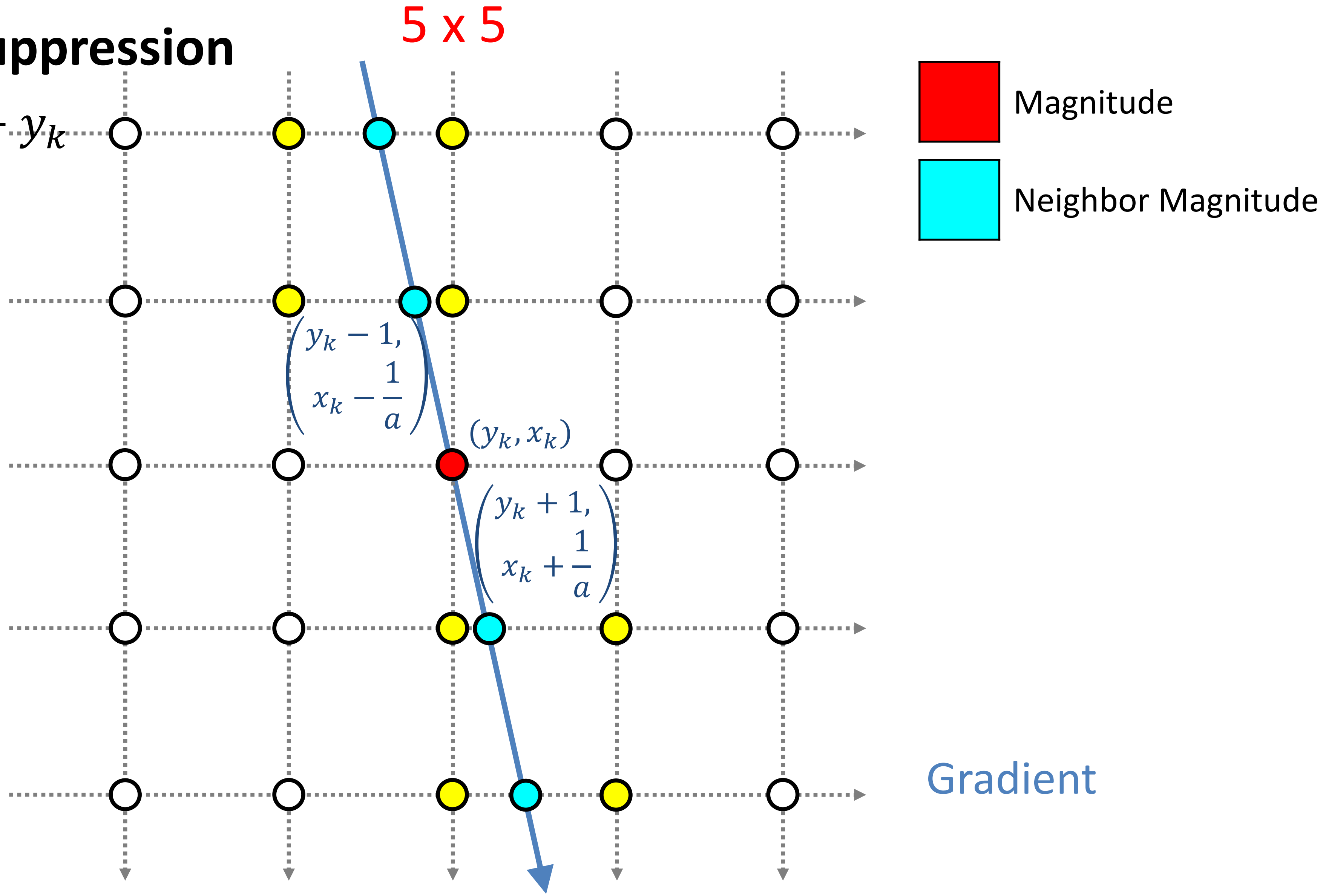
Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$


3 x 3
Magnitude
Neighbor Magnitude
$\left(y_k - 1, x_k - \frac{1}{a}\right)$
(y_k, x_k)
$\left(y_k + 1, x_k + \frac{1}{a}\right)$
Gradient

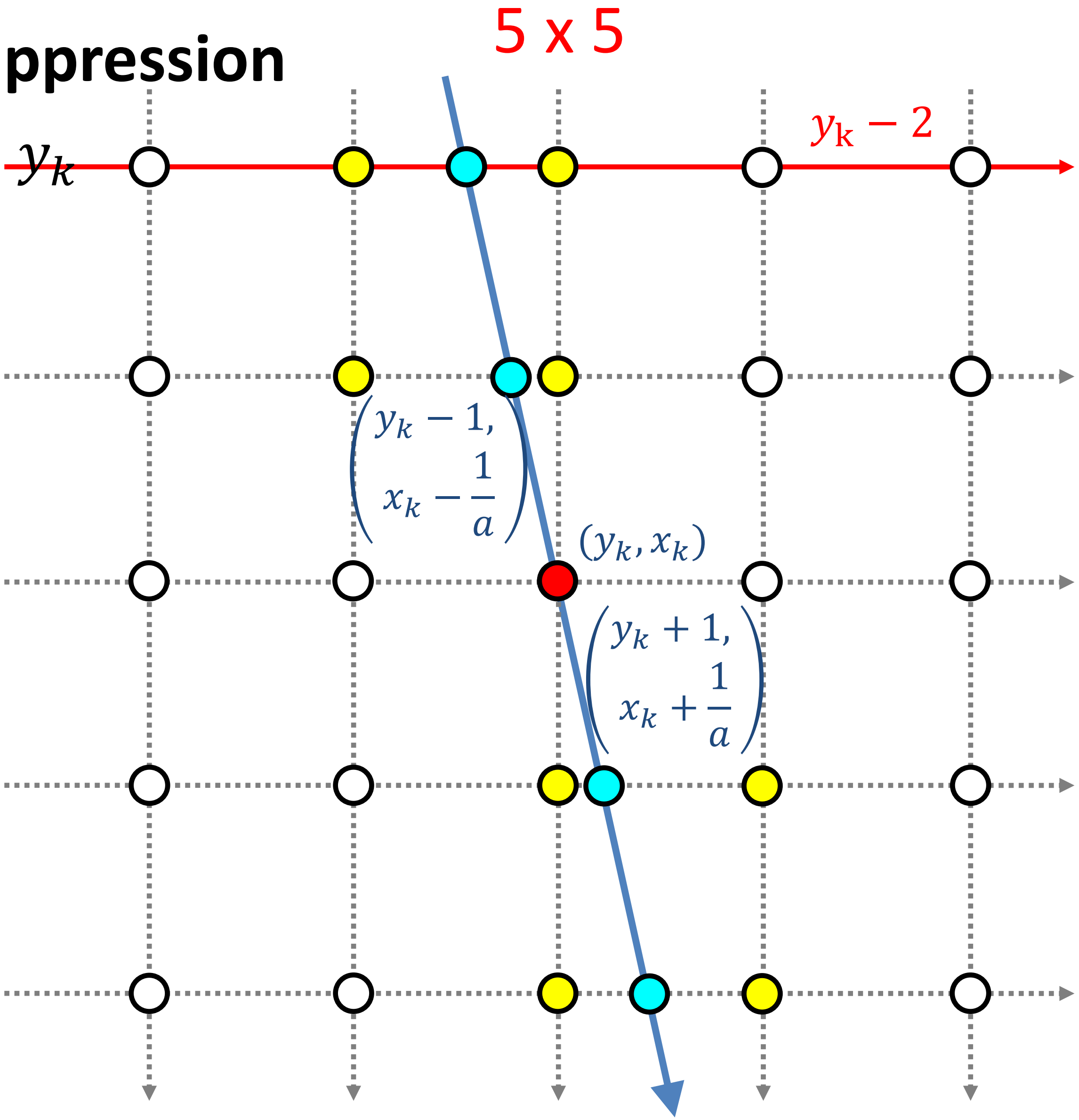
Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$


5 x 5
$\left(y_k - 1, x_k - \frac{1}{a}\right)$
(y_k, x_k)
$\left(y_k + 1, x_k + \frac{1}{a}\right)$
Magnitude
Neighbor Magnitude
Gradient

Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$


5 x 5
$y_k - 2$
$\left(y_k - 1, x_k - \frac{1}{a}\right)$
(y_k, x_k)
$\left(y_k + 1, x_k + \frac{1}{a}\right)$
Gradient

🟥 Magnitude

🟦 Neighbor Magnitude

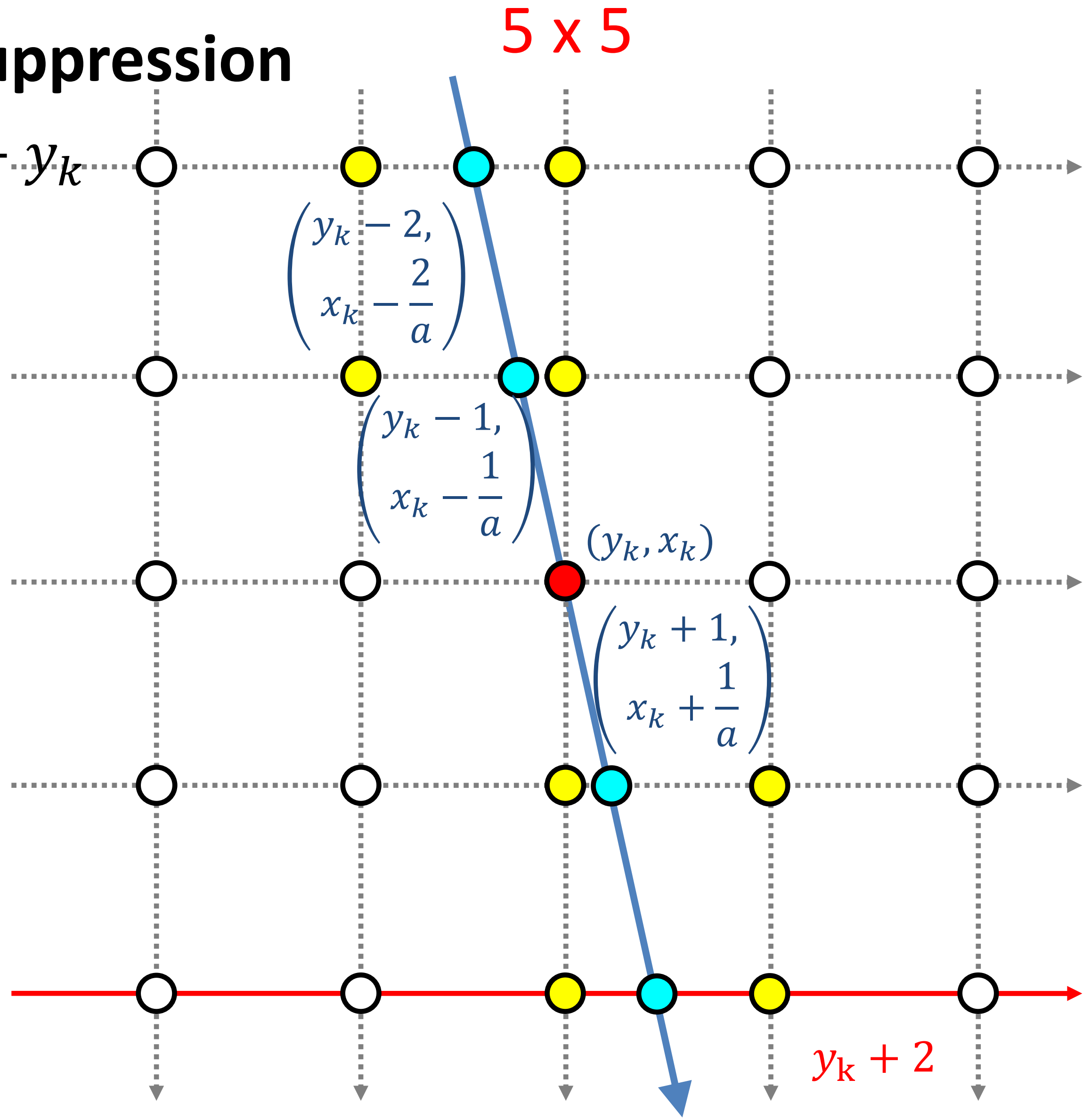
$$y_k - 2 = a(x - x_k) + y_k$$

$$-2 = a(x - x_k)$$

$$x_k - \frac{2}{a} = x$$

Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$


5 x 5
$\begin{pmatrix} y_k - 2, \\ x_k - \frac{2}{a} \end{pmatrix}$
$\begin{pmatrix} y_k - 1, \\ x_k - \frac{1}{a} \end{pmatrix}$
(y_k, x_k)
$\begin{pmatrix} y_k + 1, \\ x_k + \frac{1}{a} \end{pmatrix}$
$y_k + 2$
Gradient

🟥 Magnitude

🟦 Neighbor Magnitude

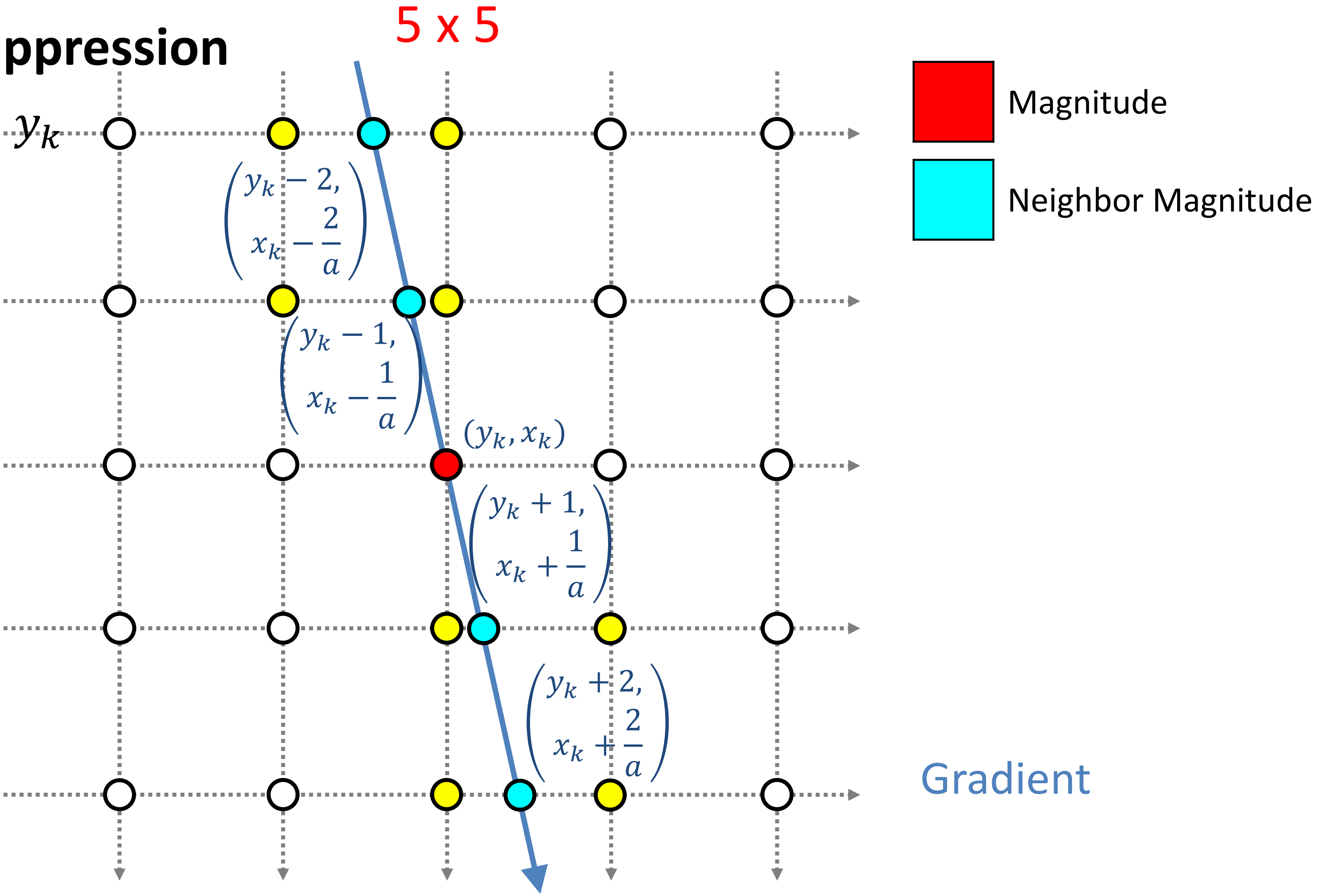
$$y_k + 2 = a(x - x_k) + y_k$$

$$2 = a(x - x_k)$$

$$x_k + \frac{2}{a} = x$$

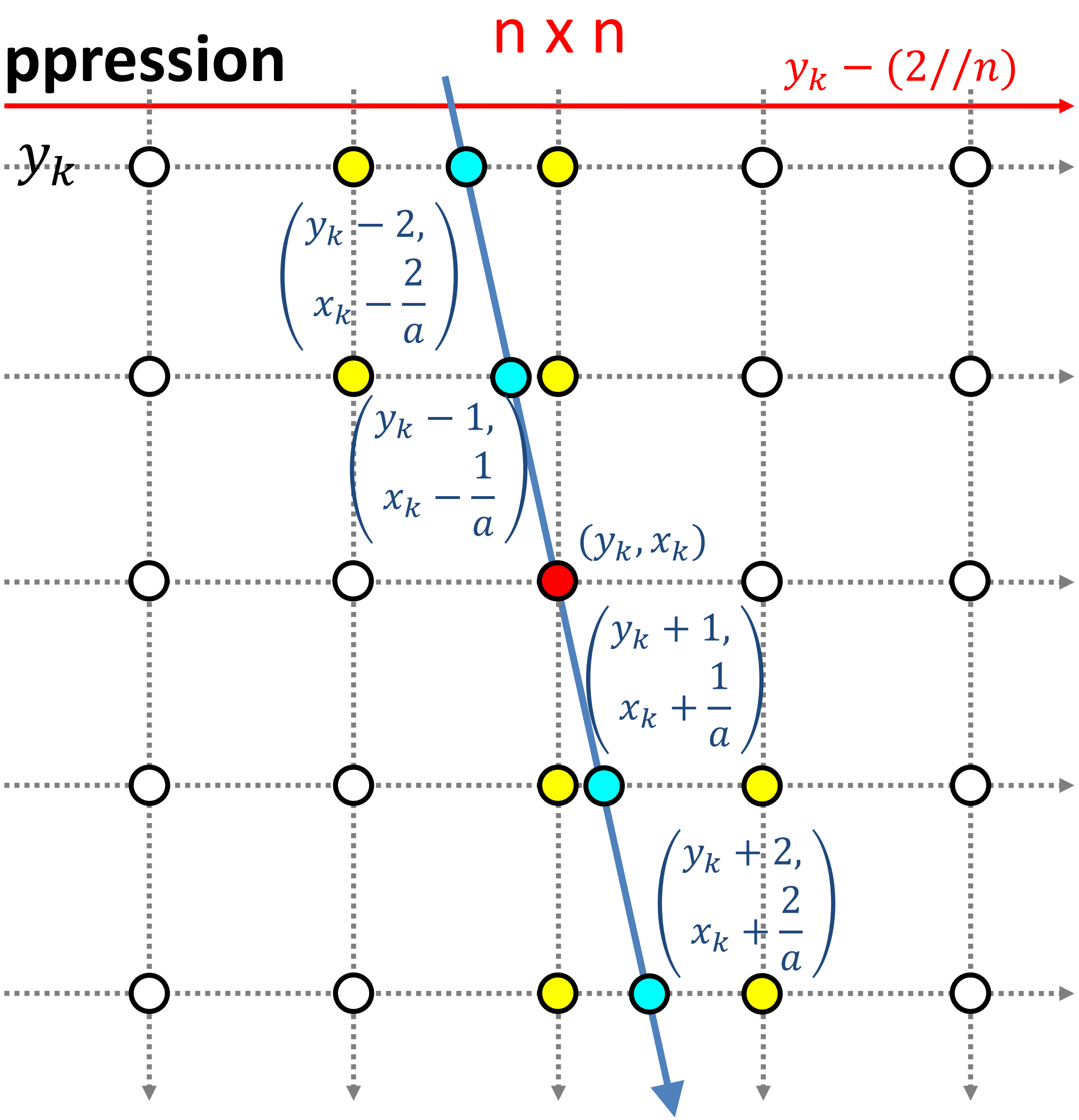
Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$


5 x 5
$\left(y_k - 2, x_k - \frac{2}{a}\right)$
$\left(y_k - 1, x_k - \frac{1}{a}\right)$
(y_k, x_k)
$\left(y_k + 1, x_k + \frac{1}{a}\right)$
$\left(y_k + 2, x_k + \frac{2}{a}\right)$
Magnitude
Neighbor Magnitude
Gradient

Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$


n x n
$y_k - (2//n)$
$\left(\begin{matrix} y_k - 2, \\ x_k - \frac{2}{a} \end{matrix}\right)$
$\left(\begin{matrix} y_k - 1, \\ x_k - \frac{1}{a} \end{matrix}\right)$
(y_k, x_k)
$\left(\begin{matrix} y_k + 1, \\ x_k + \frac{1}{a} \end{matrix}\right)$
$\left(\begin{matrix} y_k + 2, \\ x_k + \frac{2}{a} \end{matrix}\right)$

🟥 Magnitude

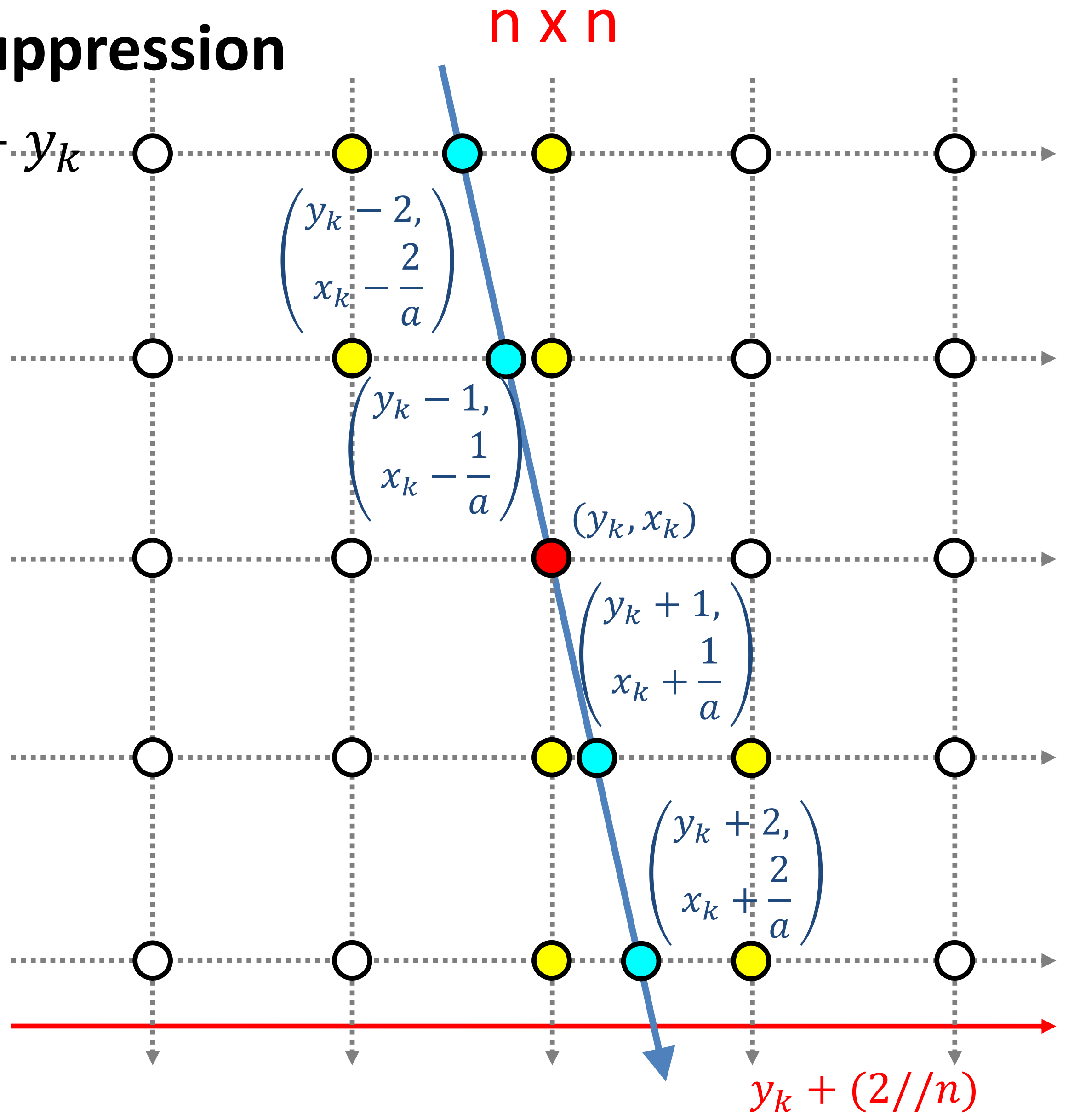
🟦 Neighbor Magnitude

$$y_k - (2//n) = a(x - x_k) + y_k$$
$$-(2//n) = a(x - x_k)$$
$$x_k - \frac{(2//n)}{a} = x$$

Gradient

Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - a(기울기) $= \frac{\nabla f_y}{\nabla f_x}$


n x n
$\left(y_k - 2, x_k - \frac{2}{a}\right)$
$\left(y_k - 1, x_k - \frac{1}{a}\right)$
(y_k, x_k)
$\left(y_k + 1, x_k + \frac{1}{a}\right)$
$\left(y_k + 2, x_k + \frac{2}{a}\right)$
$y_k + (2//n)$

🟥 Magnitude

🟦 Neighbor Magnitude

$$y_k + (2//n) = a(x - x_k) + y_k$$

$$(2//n) = a(x - x_k)$$

$$x_k + \frac{(2//n)}{a} = x$$

Gradient

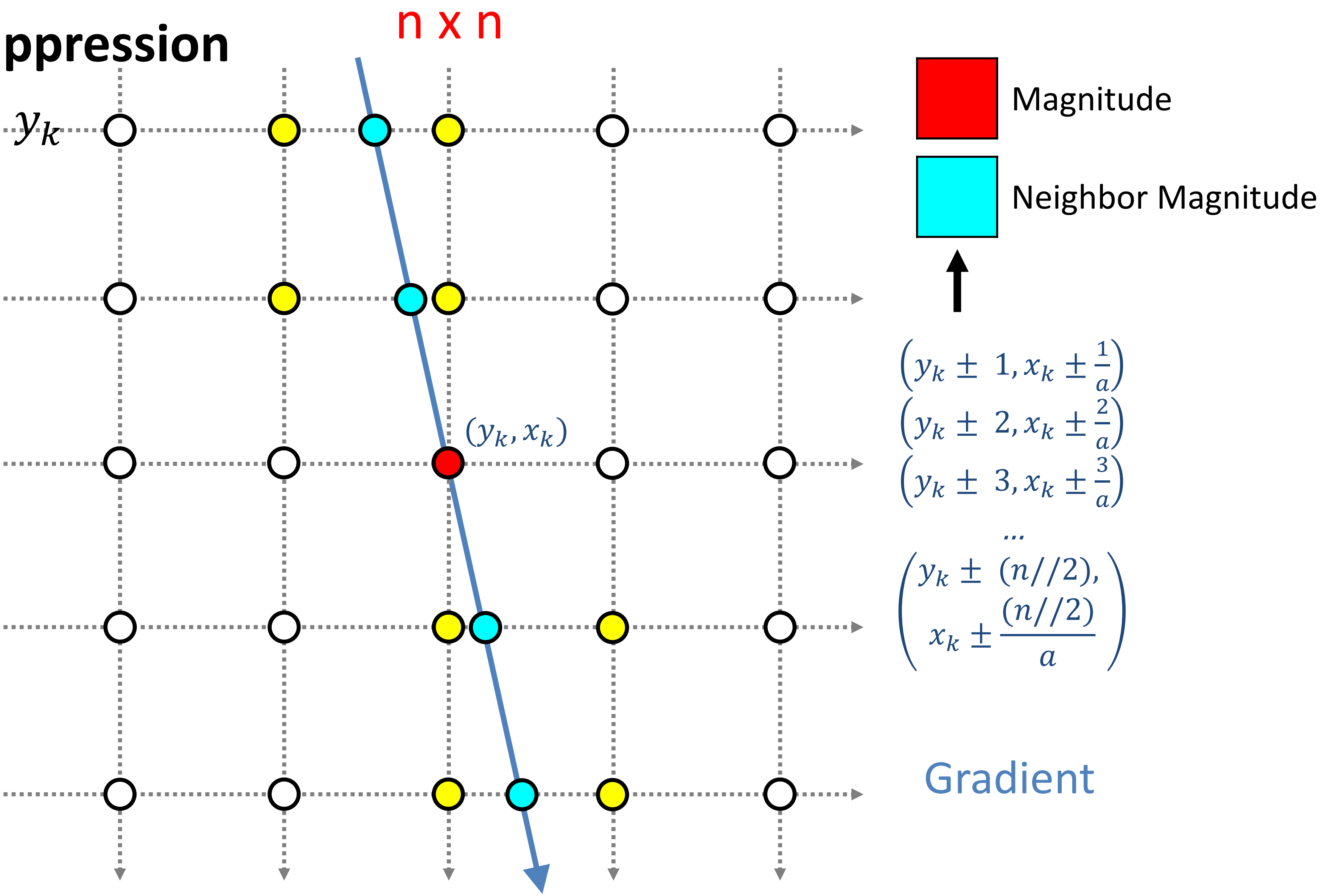
Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$

$$|\nabla f_y| > |\nabla f_x|$$

$$\left|\frac{\nabla f_y}{\nabla f_x}\right| > 1$$

$$|a| > 1$$



🟥 Magnitude

🟦 Neighbor Magnitude

$$\left(y_k \pm 1, x_k \pm \frac{1}{a}\right)$$
$$\left(y_k \pm 2, x_k \pm \frac{2}{a}\right)$$
$$\left(y_k \pm 3, x_k \pm \frac{3}{a}\right)$$
$$\dots$$
$$\left(y_k \pm (n//2), x_k \pm \frac{(n//2)}{a}\right)$$

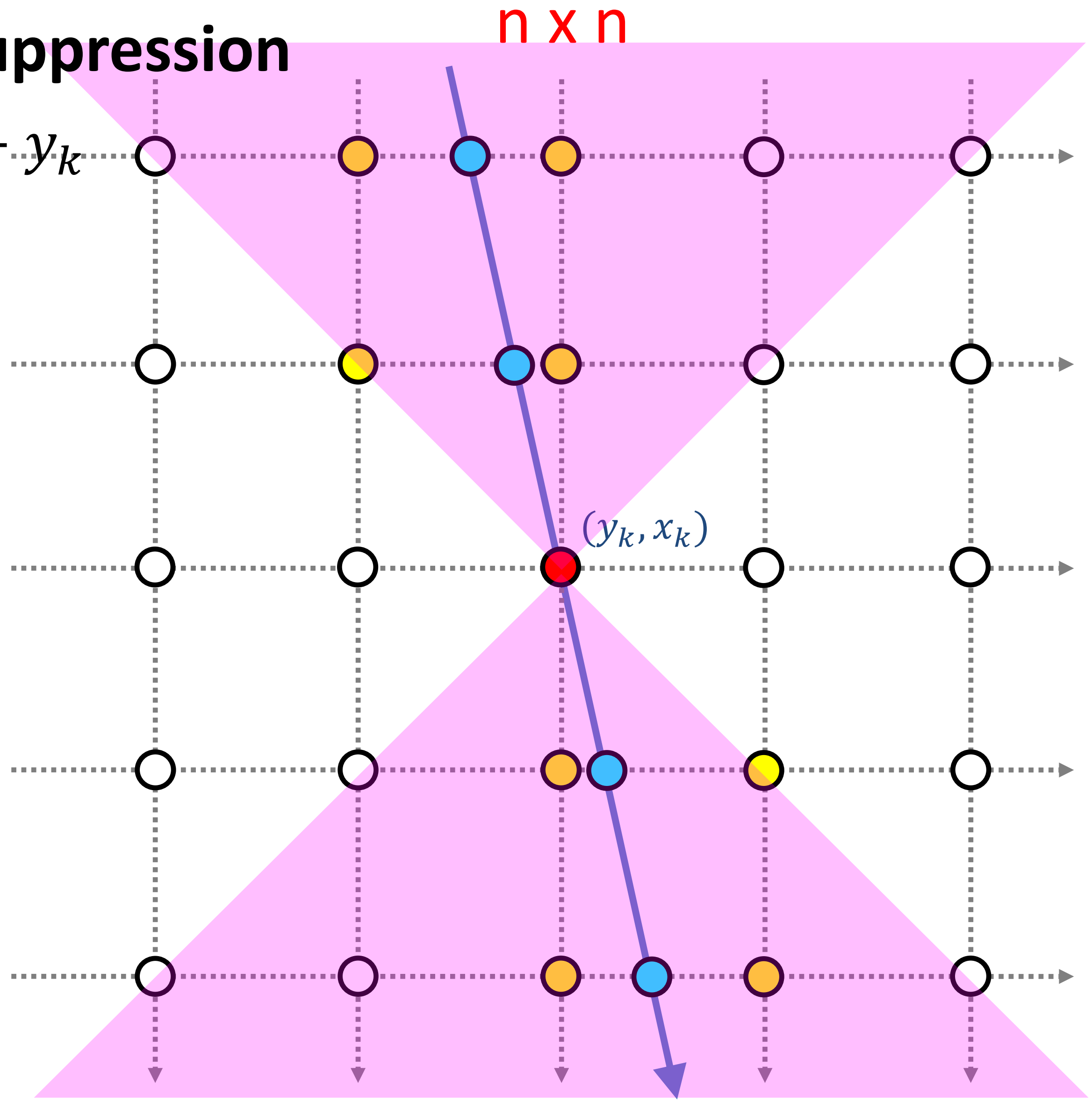
Canny edge detection

- **Non-maximum suppression**
  - $- \ y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$

$$|\nabla f_y| > |\nabla f_x|$$

$$\left|\frac{\nabla f_y}{\nabla f_x}\right| > 1$$

$$|a| > 1$$



🟥 Magnitude

🟦 Neighbor Magnitude

↑

$\left(y_k \pm 1, x_k \pm \frac{1}{a}\right)$
$\left(y_k \pm 2, x_k \pm \frac{2}{a}\right)$
$\left(y_k \pm 3, x_k \pm \frac{3}{a}\right)$
...
$\left(y_k \pm (n//2), x_k \pm \frac{(n//2)}{a}\right)$

Gradient

Canny edge detection

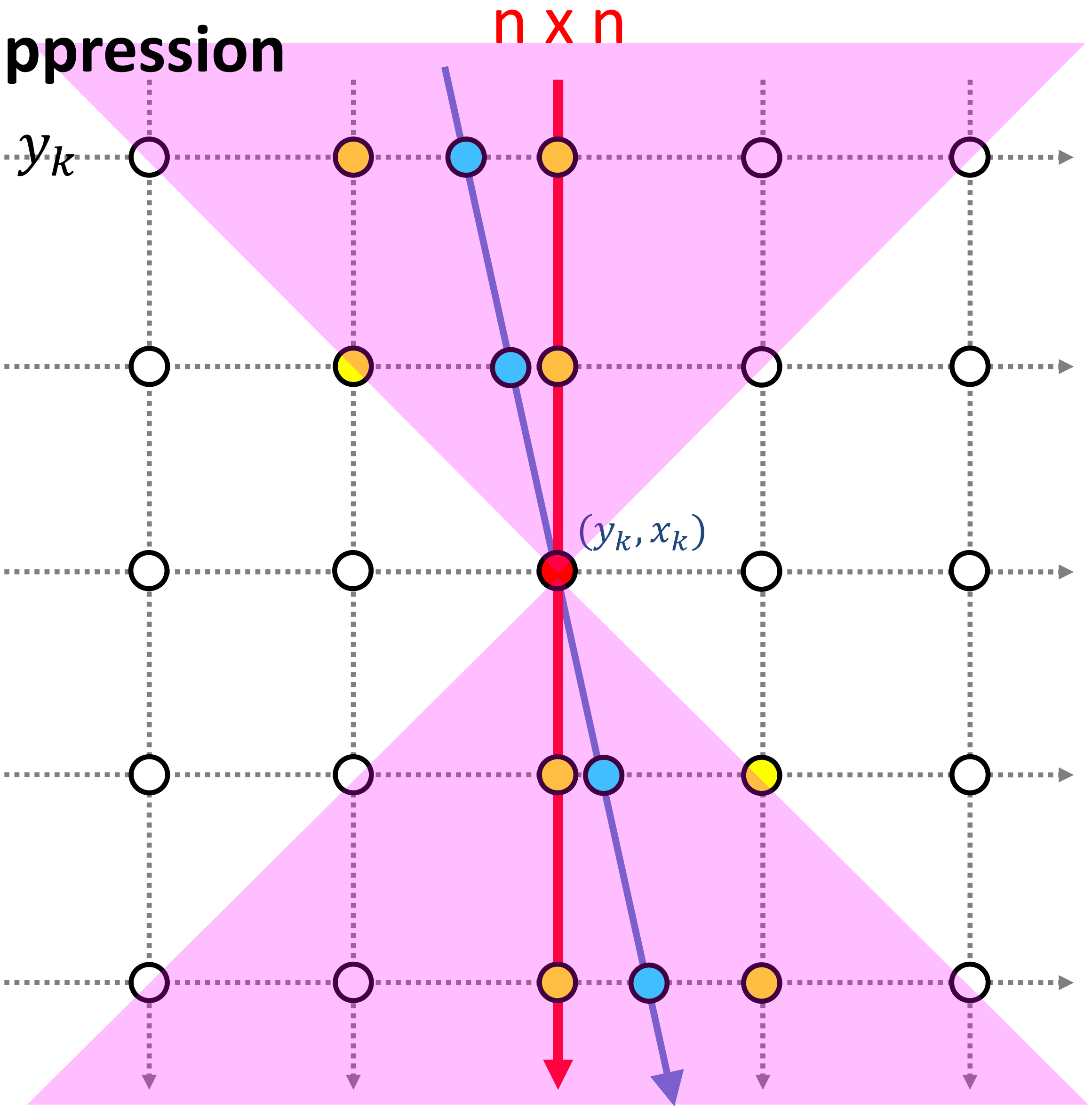
- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - a(기울기) $= \frac{\nabla f_y}{\nabla f_x}$

$$|\nabla f_y| > |\nabla f_x|$$

$$\left|\frac{\nabla f_y}{\nabla f_x}\right| > 1$$

$$|a| > 1$$

$$\nabla f_x \neq 0$$



🟥 Magnitude

🟦 Neighbor Magnitude

$$\left(y_k \pm 1, x_k \pm \frac{1}{a}\right)$$
$$\left(y_k \pm 2, x_k \pm \frac{2}{a}\right)$$
$$\left(y_k \pm 3, x_k \pm \frac{3}{a}\right)$$
$$\dots$$
$$\left(\begin{array}{c} y_k \pm (n//2), \\ x_k \pm \frac{(n//2)}{a} \end{array}\right)$$

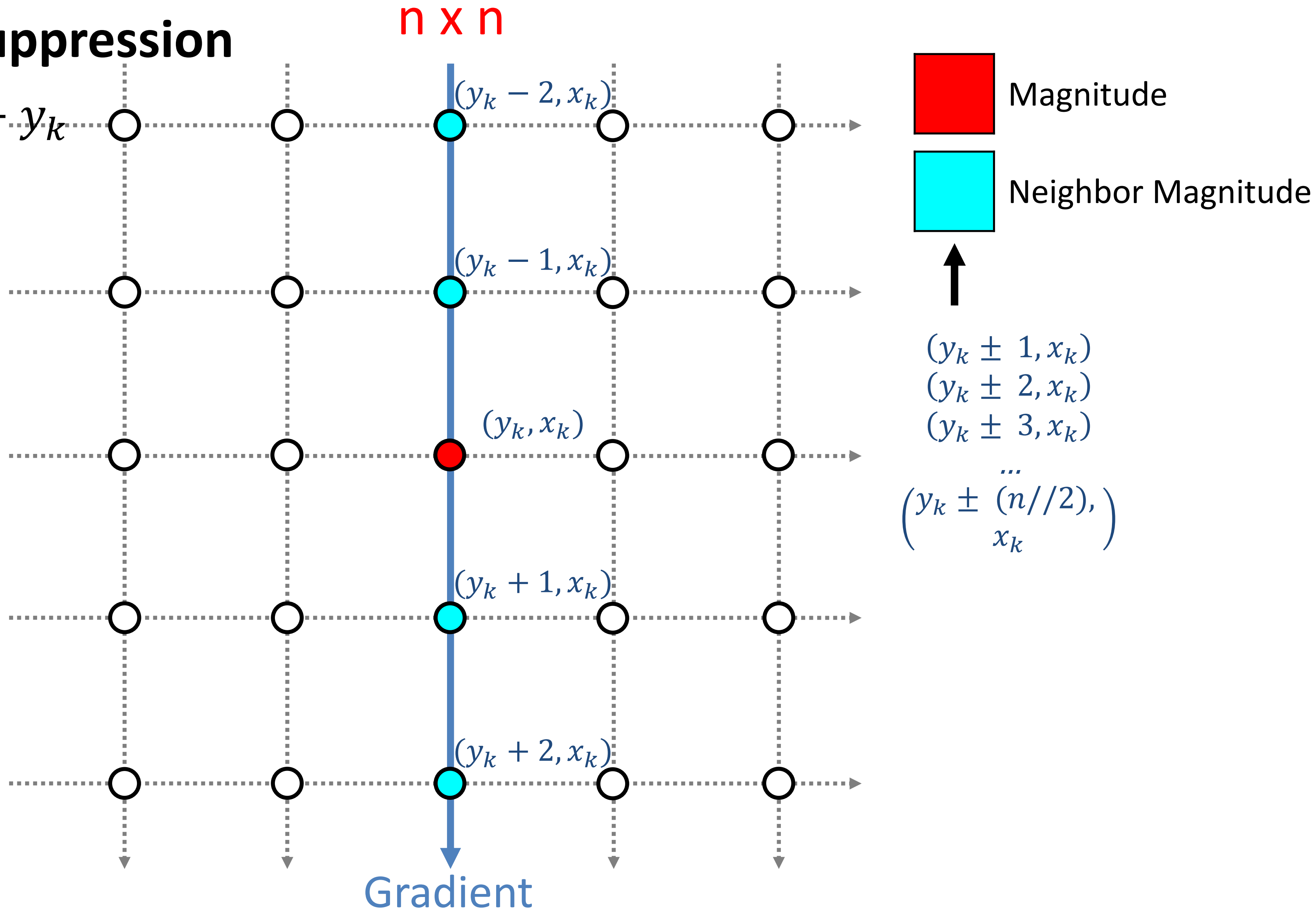
Gradient

Canny edge detection

- **Non-maximum suppression**
  - $y = a(x - x_k) + y_k$
    - $a(\text{기울기}) = \frac{\nabla f_y}{\nabla f_x}$

$y = x_k$

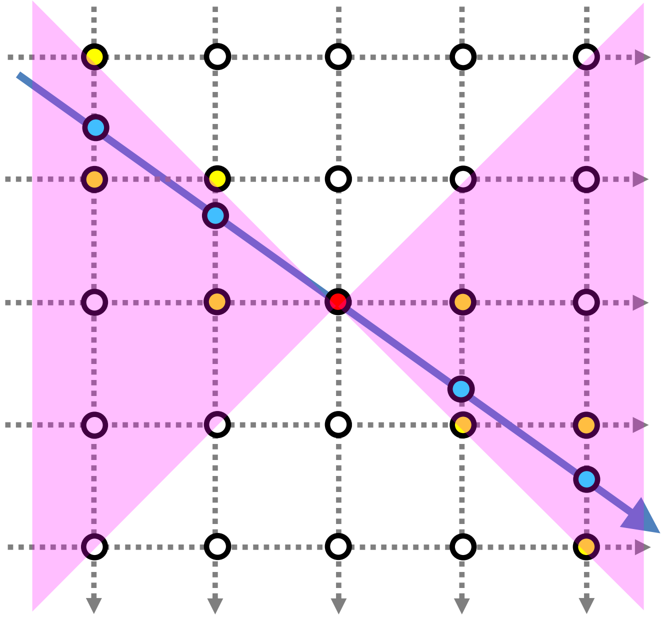
$\nabla f_x = 0$



$(y_k \pm 1, x_k)$
$(y_k \pm 2, x_k)$
$(y_k \pm 3, x_k)$
...
$\begin{pmatrix} y_k \pm (n//2), \\ x_k \end{pmatrix}$

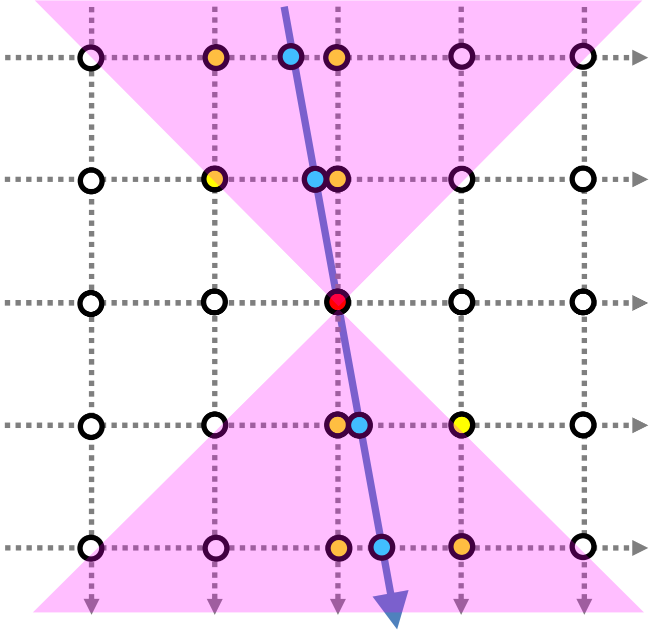
Canny edge detection

- **Non-maximum suppression**
  - 3 Case ($a = \frac{\nabla f_y}{\nabla f_x}$)



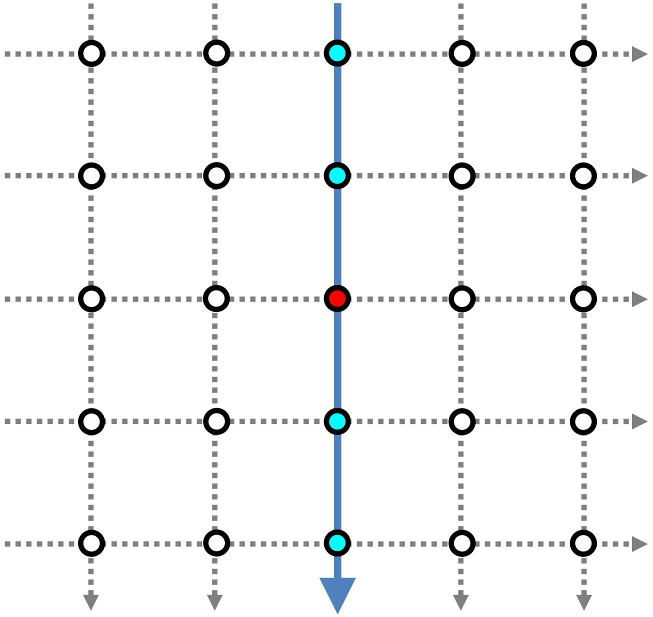
$|a| < 1$

$$\begin{pmatrix} y_k \pm a \times (i\ //2), \\ x_k \pm\ (i\ //2) \end{pmatrix}$$



$|a| \geq 1$
$(|a| \neq \infty)$

$$\begin{pmatrix} y_k \pm (i\ //2), \\ x_k \pm \dfrac{(i\ //2)}{a} \end{pmatrix}$$

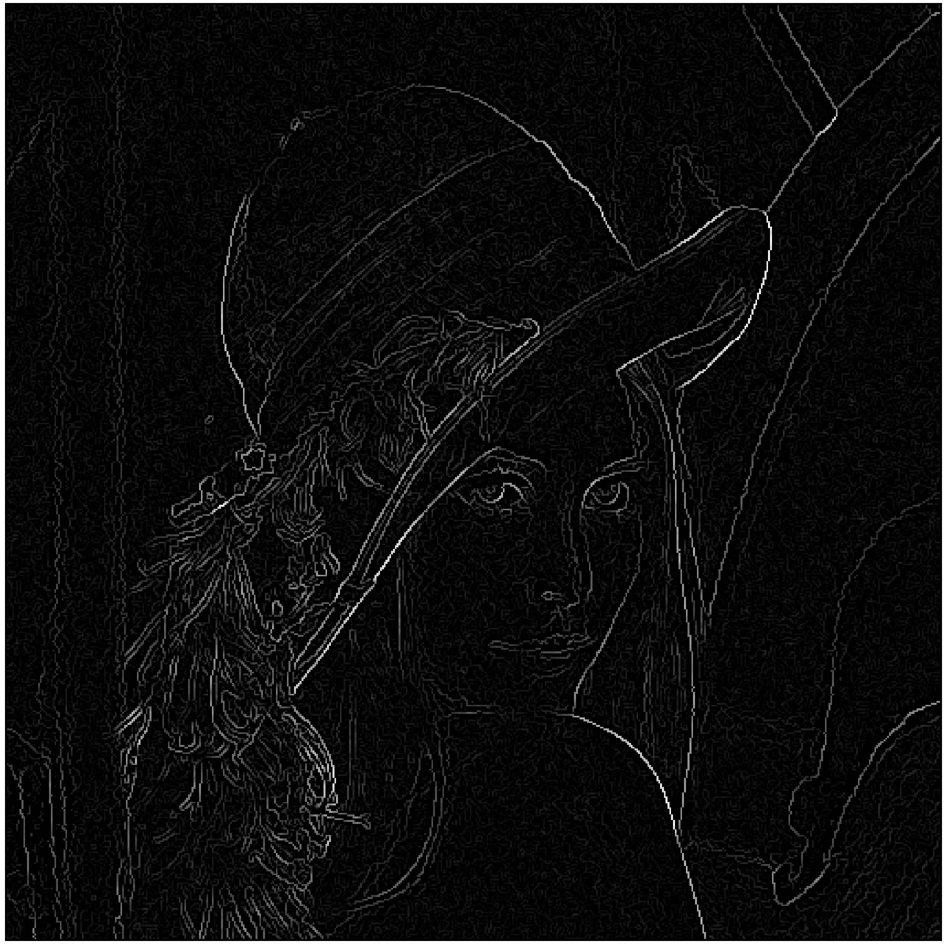


$\nabla f_x = 0$
$(|a| = \infty)$

$$\begin{pmatrix} y_k \pm\ (i\ //2), \\ x_k \end{pmatrix}$$

Canny edge detection

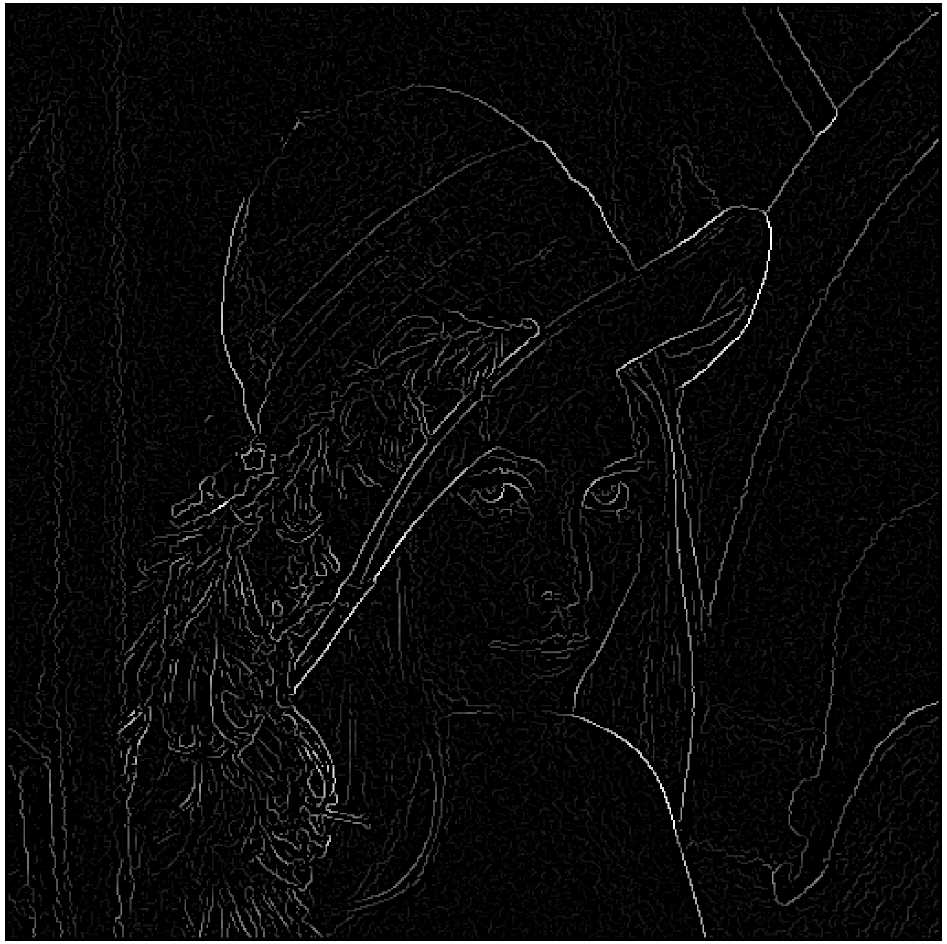
- **Non-maximum suppression**



3 x 3

Canny edge detection

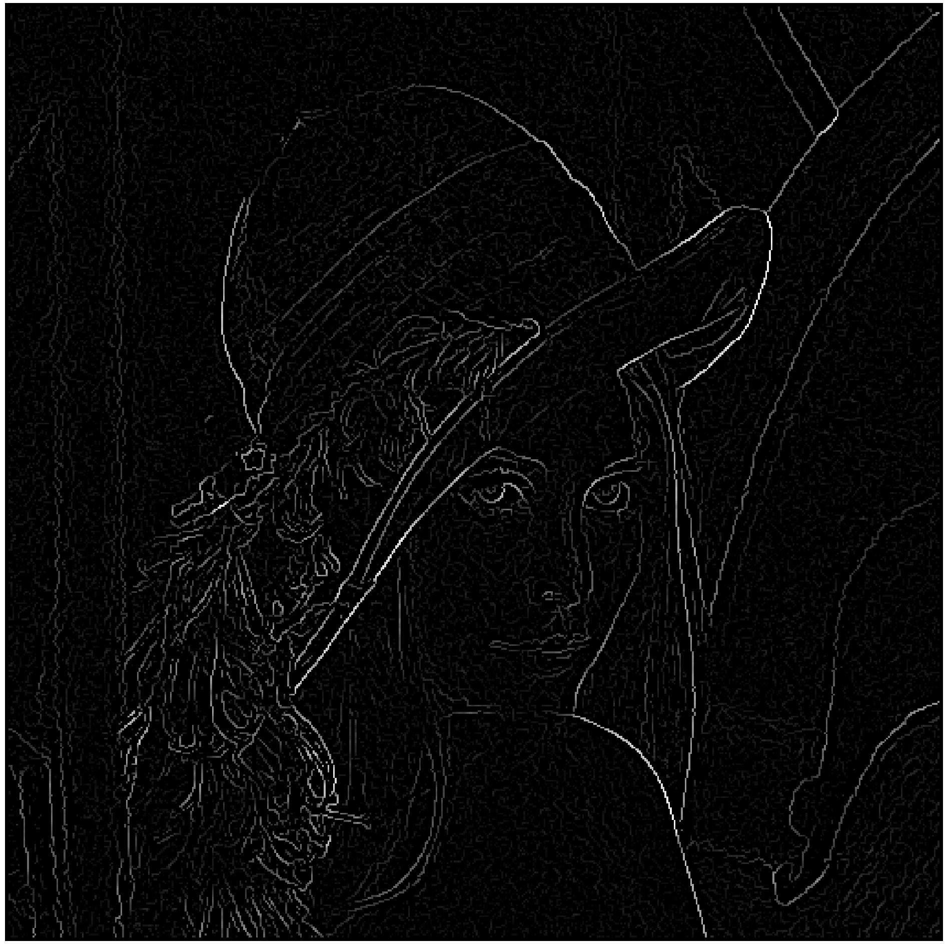
- **Non-maximum suppression**



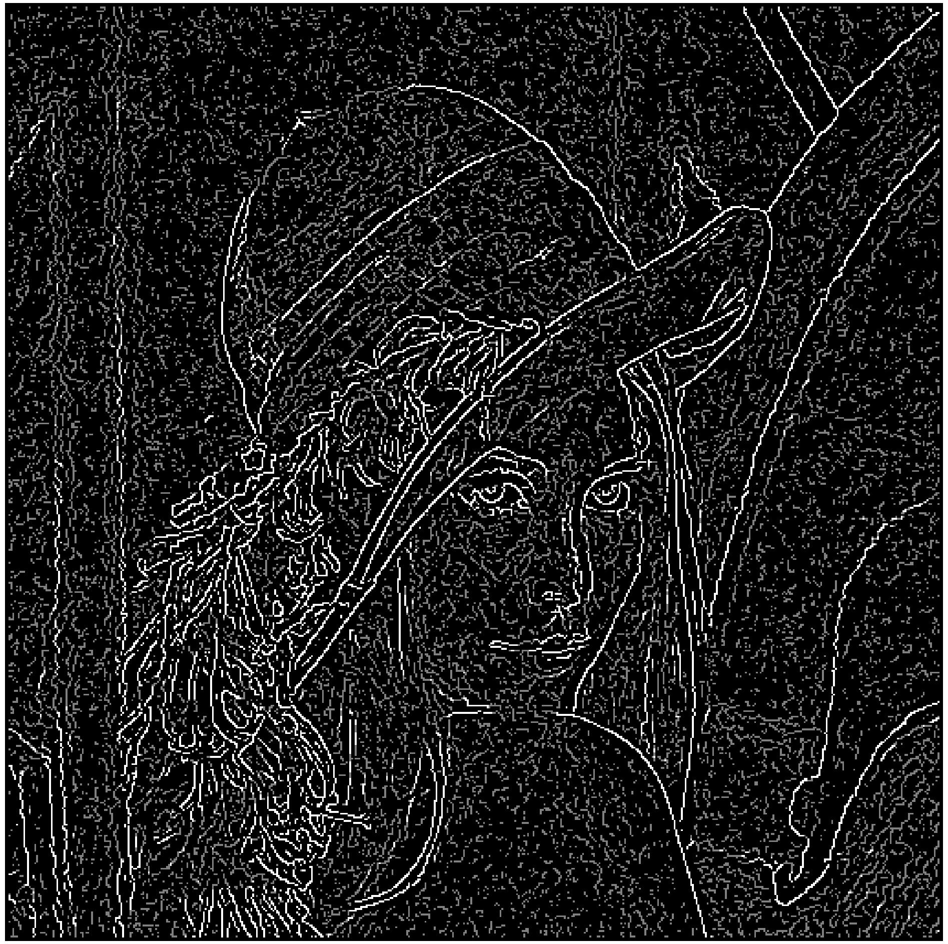
5 x 5

Canny edge detection

- **Double thresholding** (T_L, T_H)



Non−maximum suppression
(5 x 5)



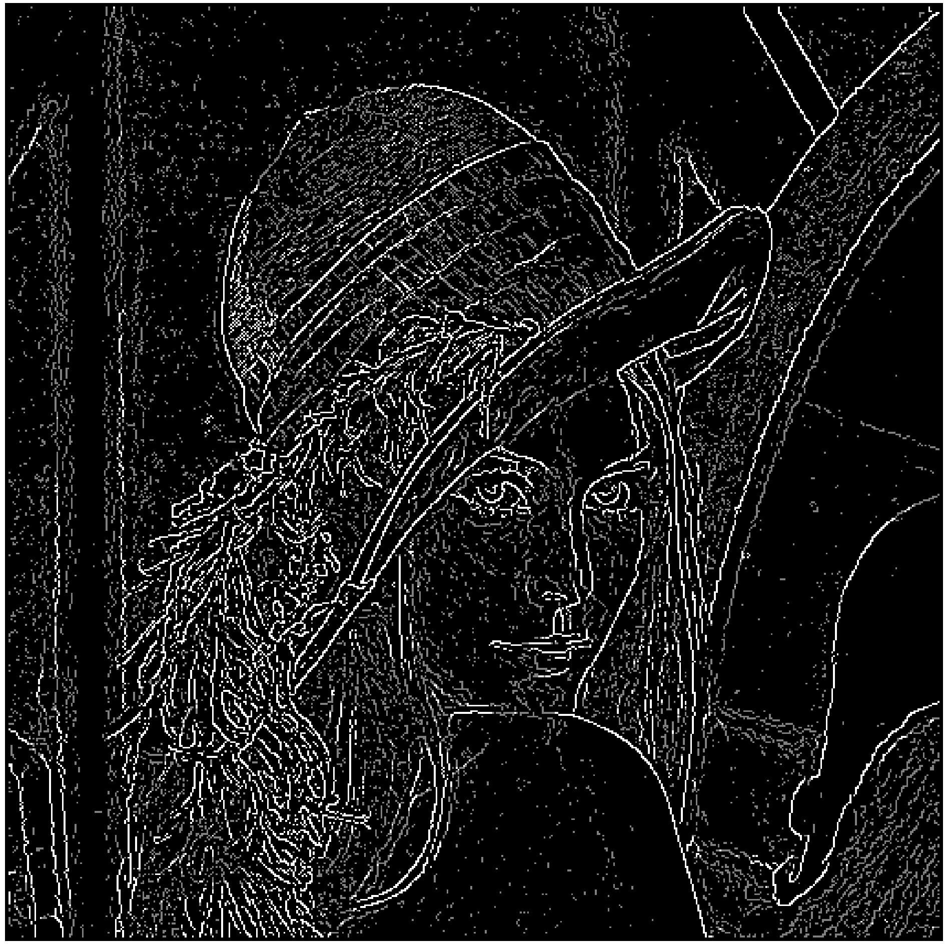
Double thresholding

Canny edge detection

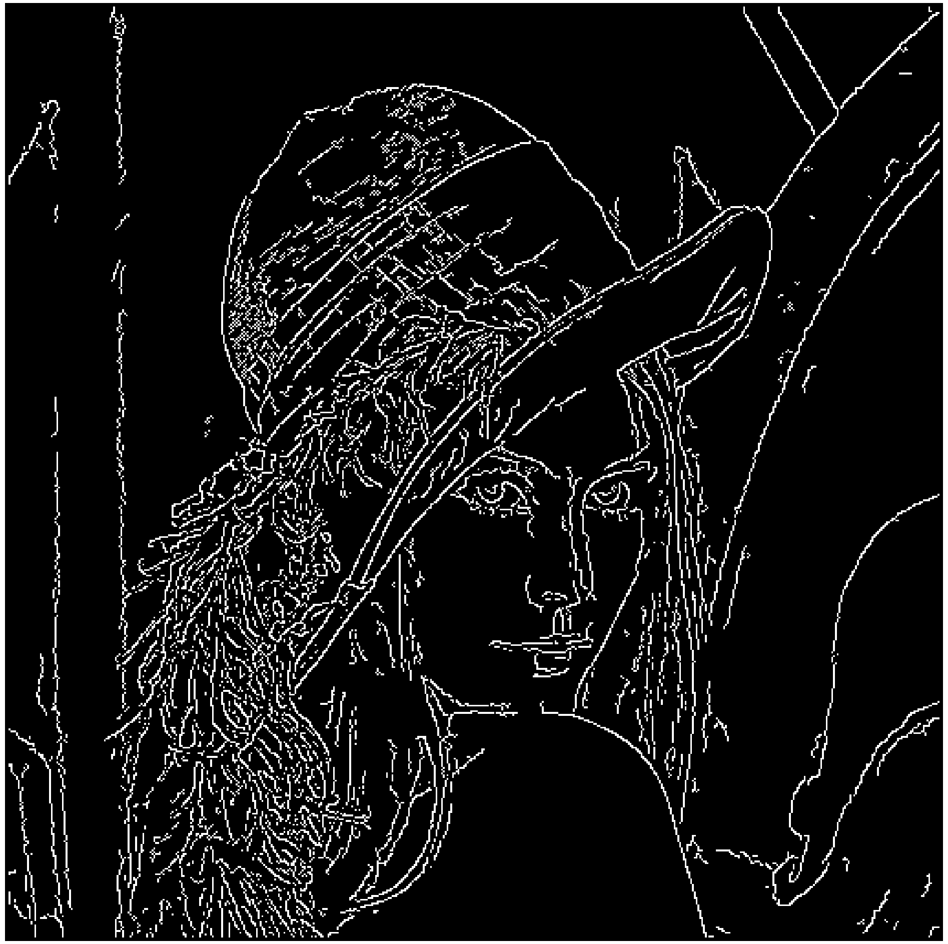
- **Double thresholding** (T_L, T_H)
  - If $M(y,x) > T_H$, then (y,x) is a strong edge
    - $\mathrm{dst}(y,x) = 255$
  - If $M(y,x) < T_\mathrm{L}$, then (y,x) is NOT an edge
    - $\mathrm{dst}(y,x) = 0$
  - If $T_L \leq M(y,x) \leq T_H$, then (y,x) is a weak edge
    - $\mathrm{dst}(y,x) = 128$
    - 좌표를 저장함

Canny edge detection

- **Determine edge**



Double thresholding



Determine edge

Canny edge detection

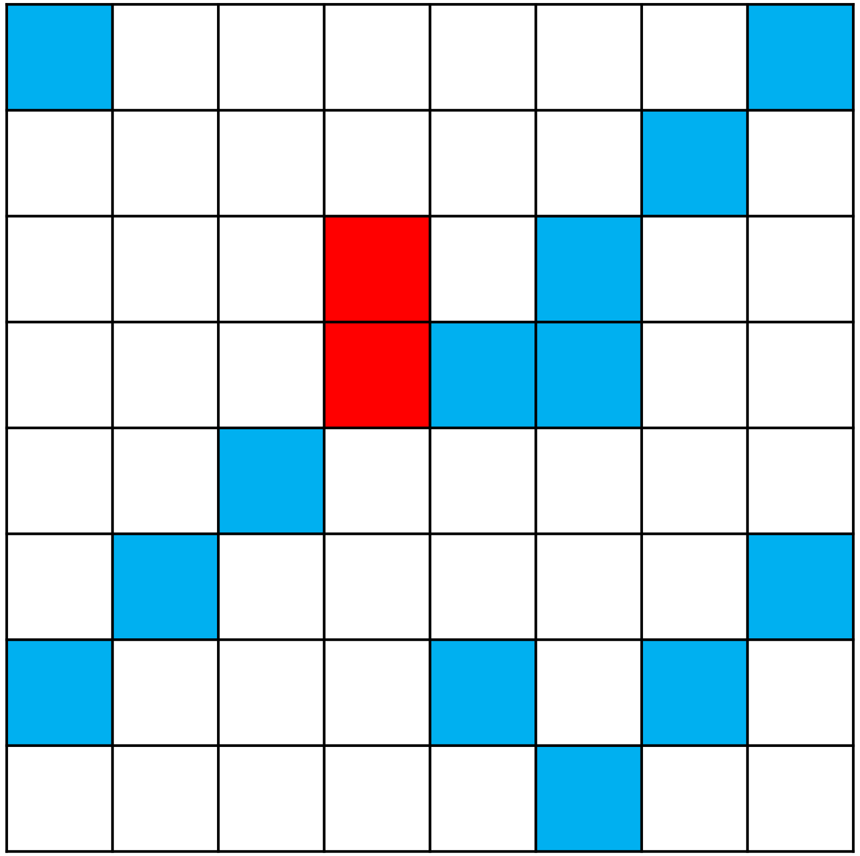
- **Determine edge**
  - Connect: []

🟥 Strong Edge

🟦 Weak Edge

🟪 Now

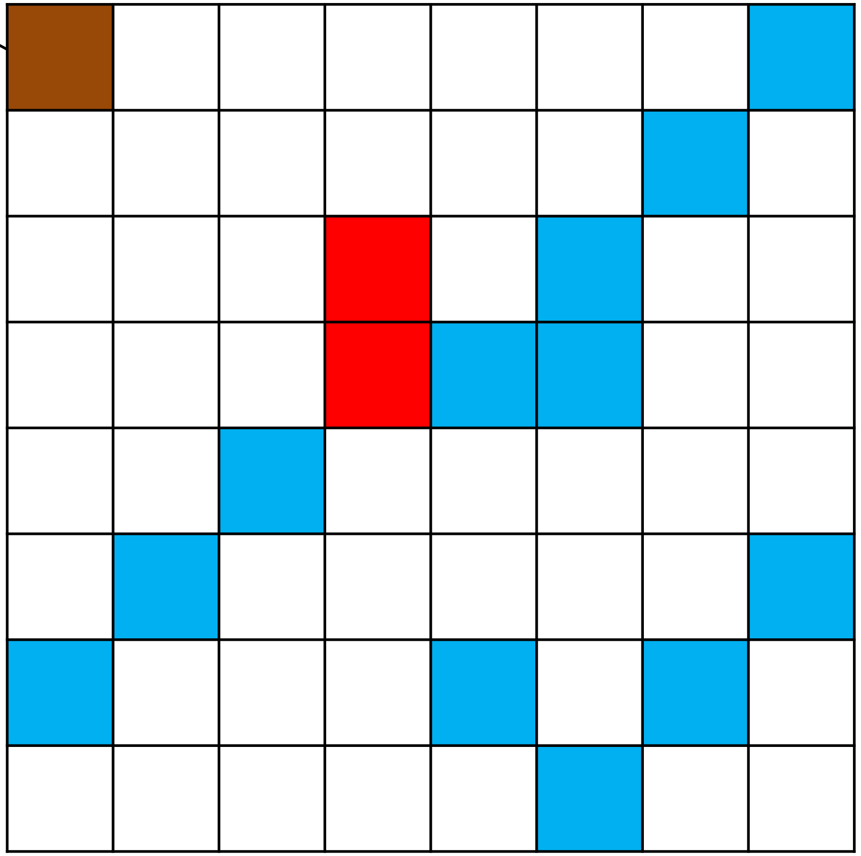
🟫 Queue



Magnitude

Canny edge detection

- **Determine edge**
  - Connect: [(0, 0)]

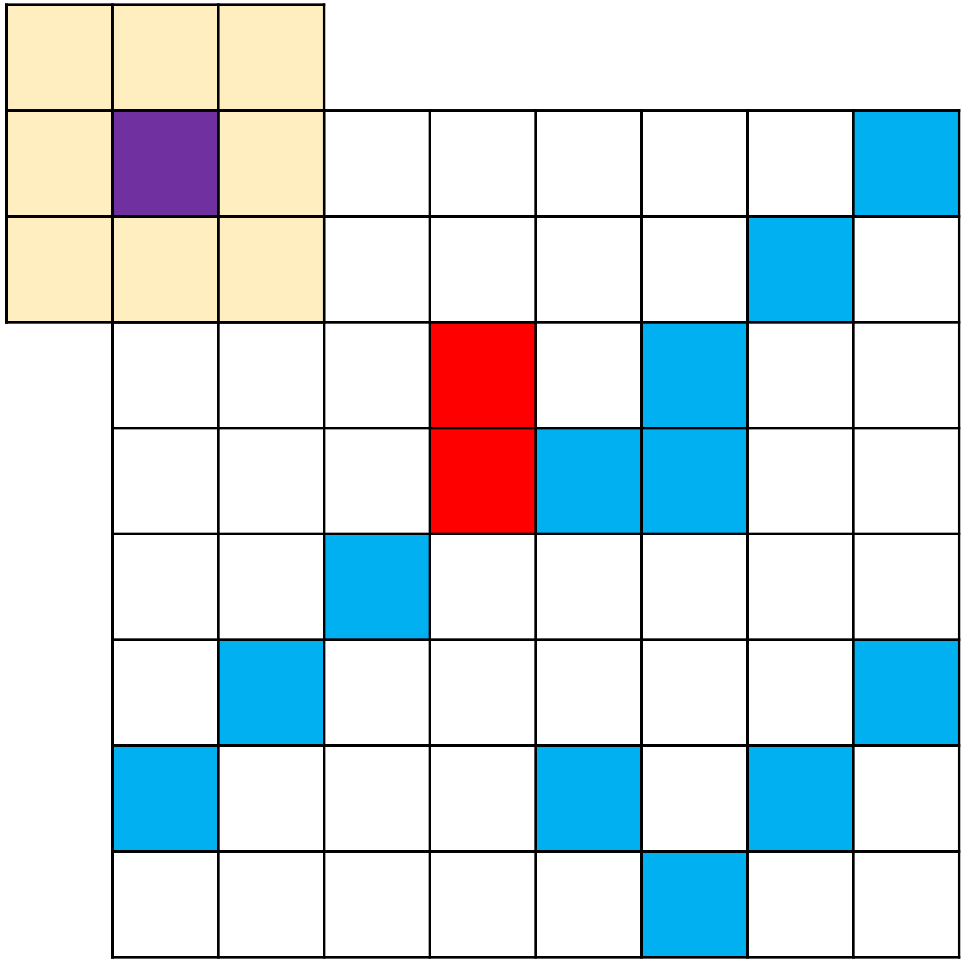


Magnitude

- 🟥 Strong Edge
- 🟦 Weak Edge
- 🟪 Now
- 🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [(0, 0)]

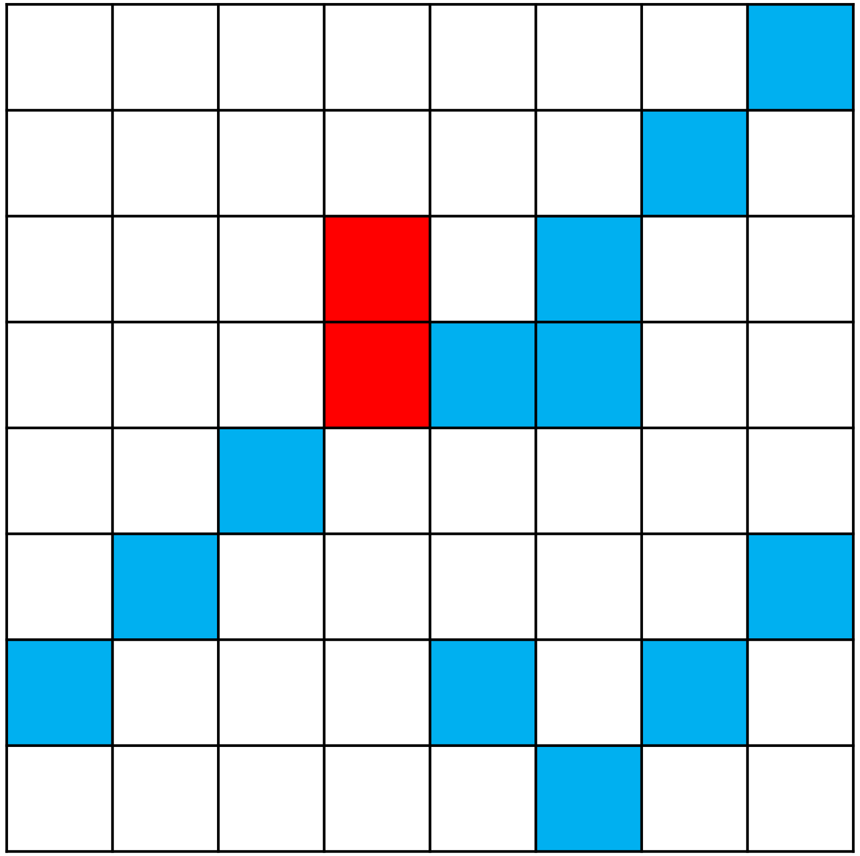


Magnitude

- 🟥 Strong Edge
- 🟦 Weak Edge
- 🟪 Now
- 🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: []



Magnitude

🟥 Strong Edge

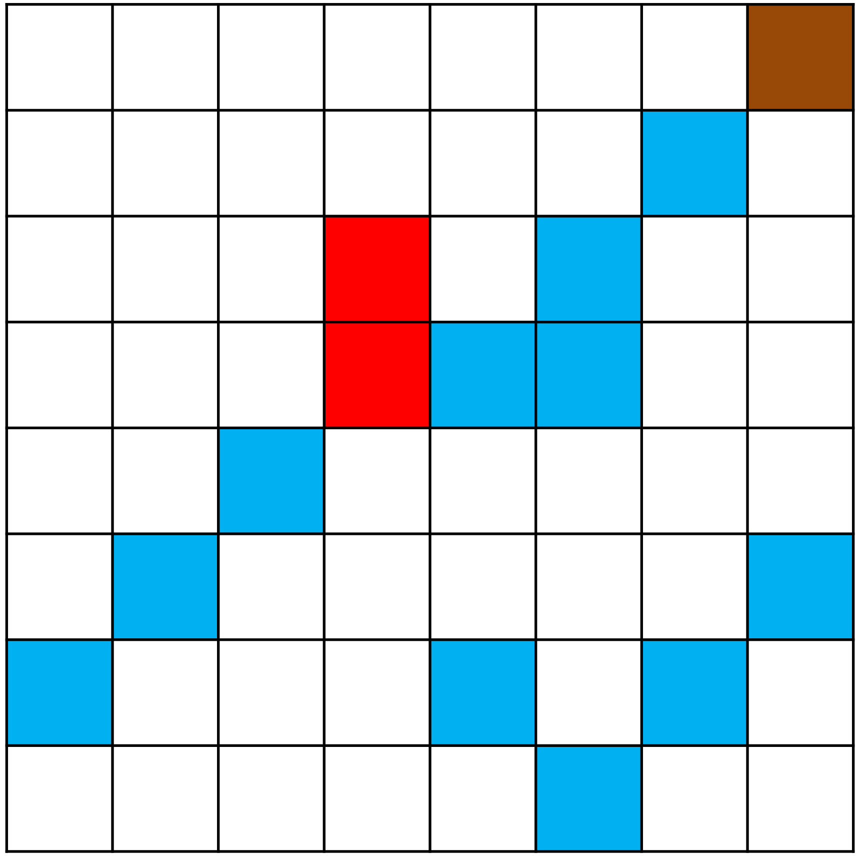
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [(0, 7)]



Magnitude

🟥 Strong Edge

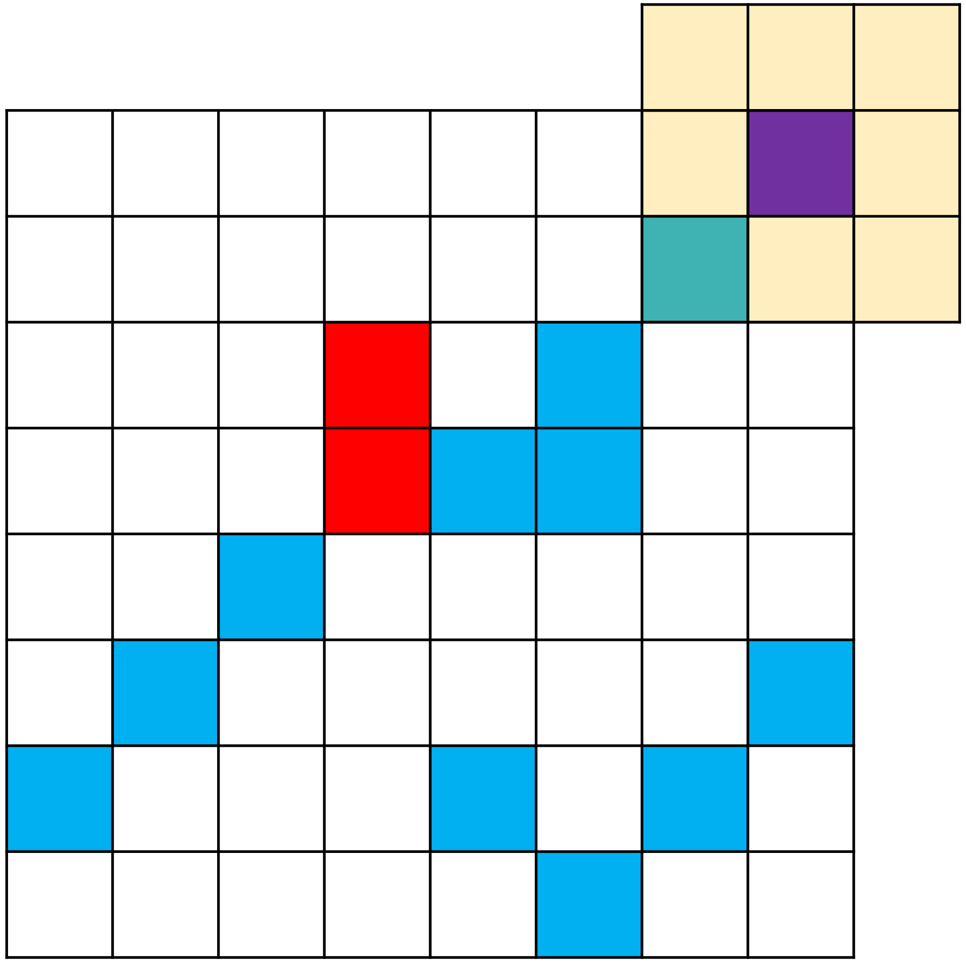
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [(0, 7)]



Magnitude

🟥 Strong Edge

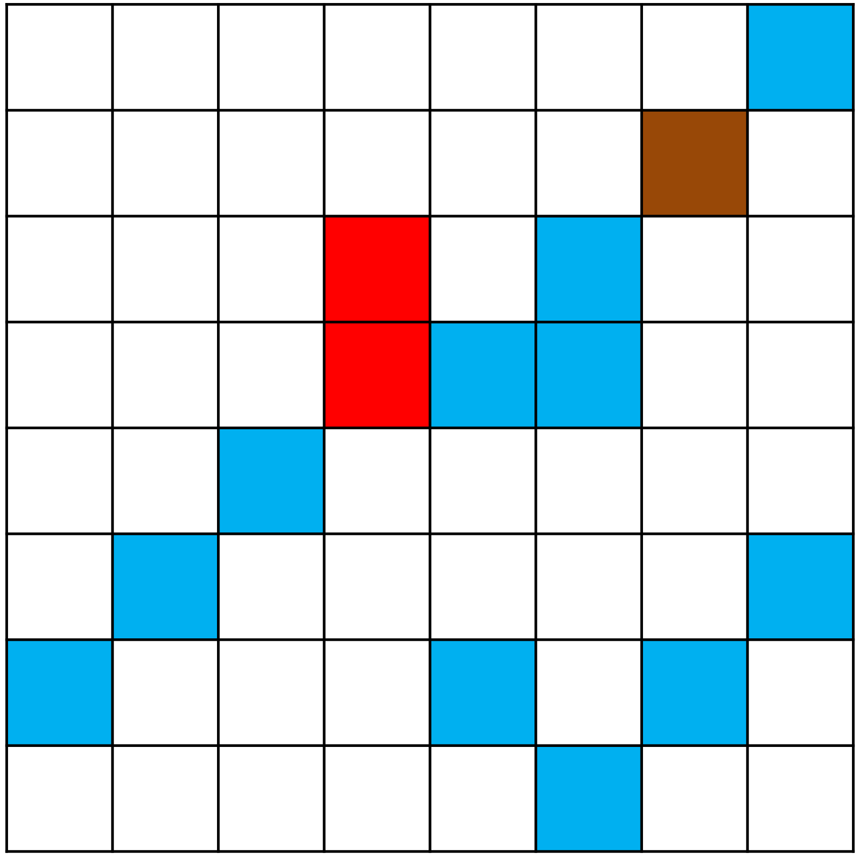
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [(0, 7), (1, 6)]



Magnitude

🟥 Strong Edge

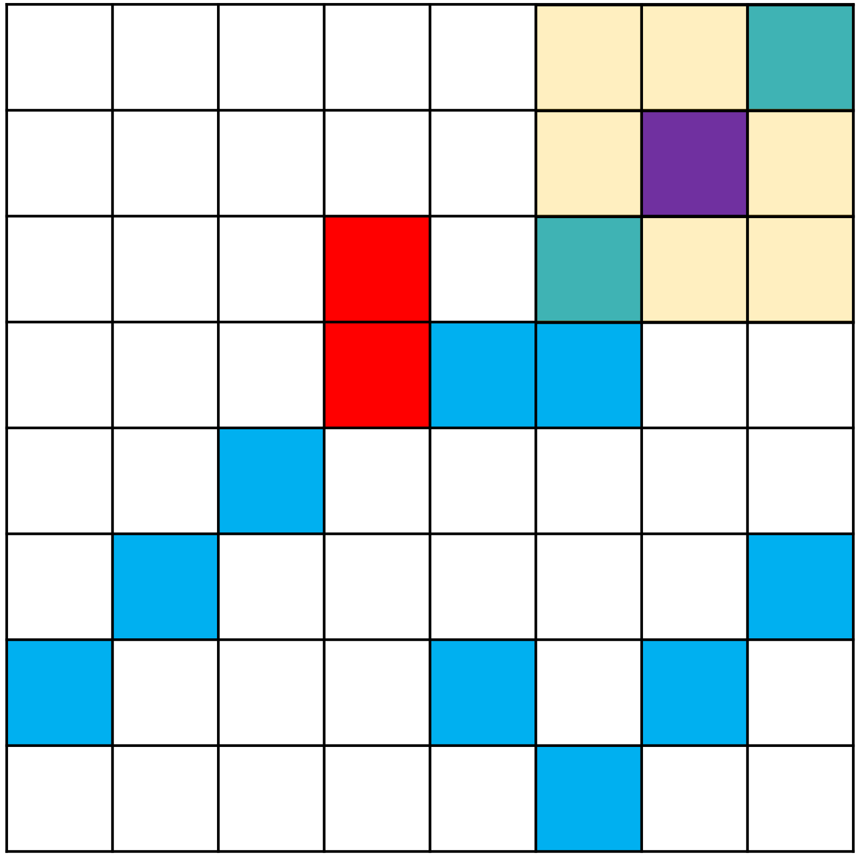
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [(0, 7), (1, 6)]



Magnitude

🟥 Strong Edge

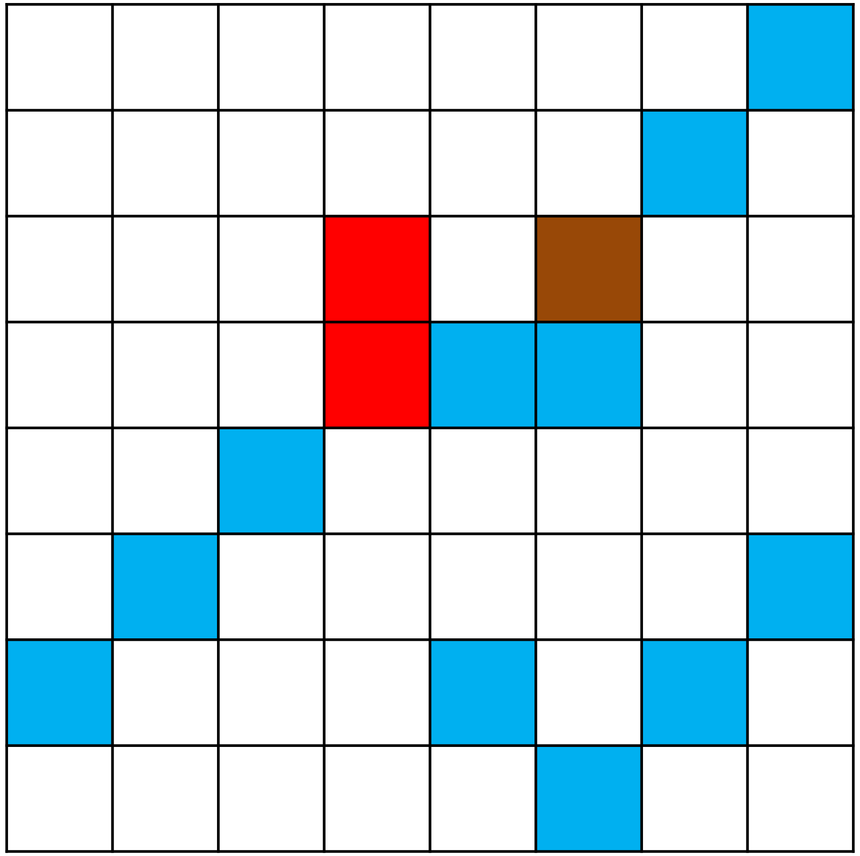
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [(0, 7), (1, 6), (2, 5)]



Magnitude

🟥 Strong Edge

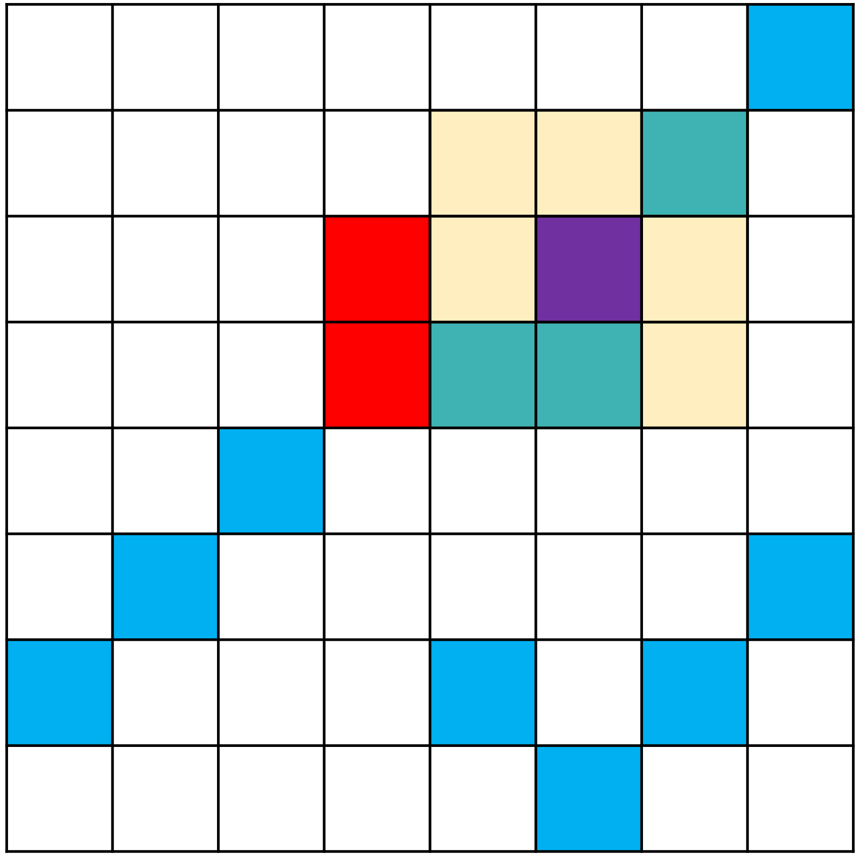
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [(0, 7), (1, 6), (2, 5)]



Magnitude

🟥 Strong Edge

🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

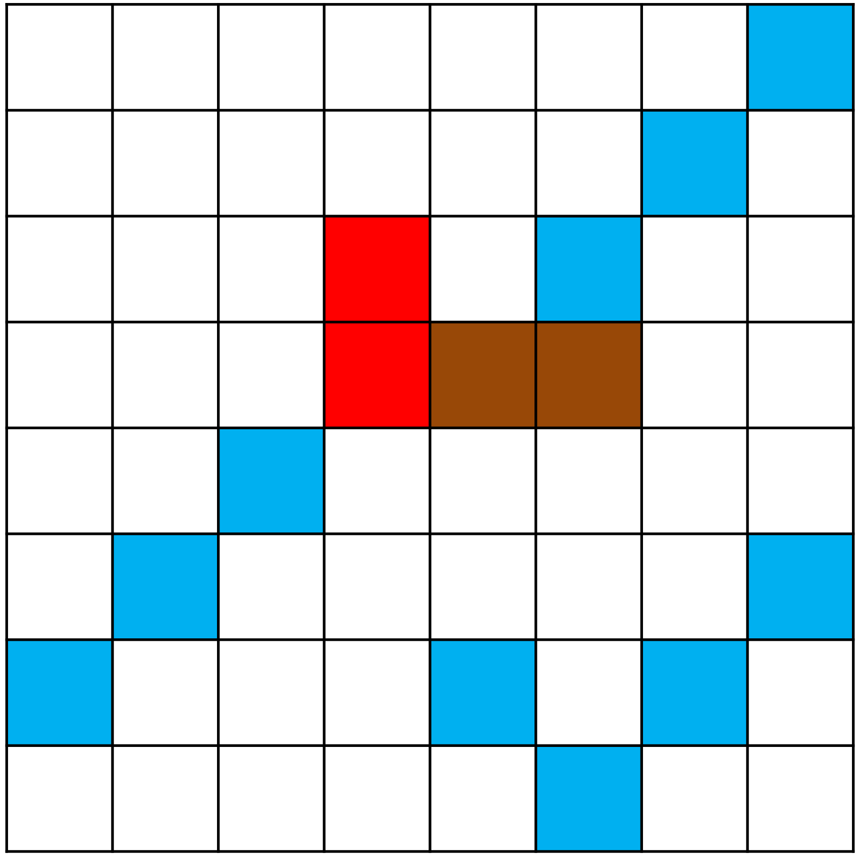
- **Determine edge**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4)]

🟥 Strong Edge

🟦 Weak Edge

🟪 Now

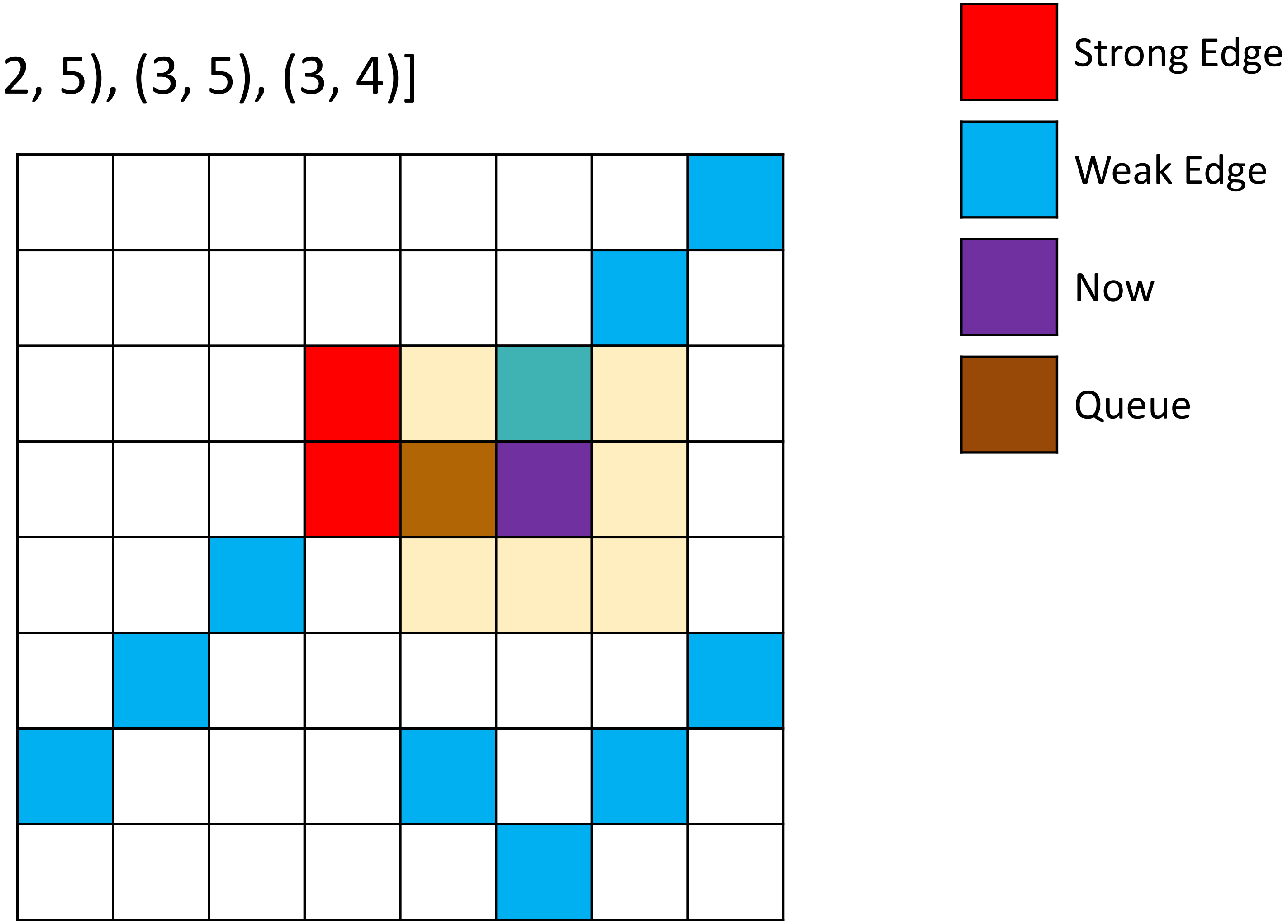
🟫 Queue



Magnitude

Canny edge detection

- **Determine edge**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4)]



🟥 Strong Edge

🟦 Weak Edge

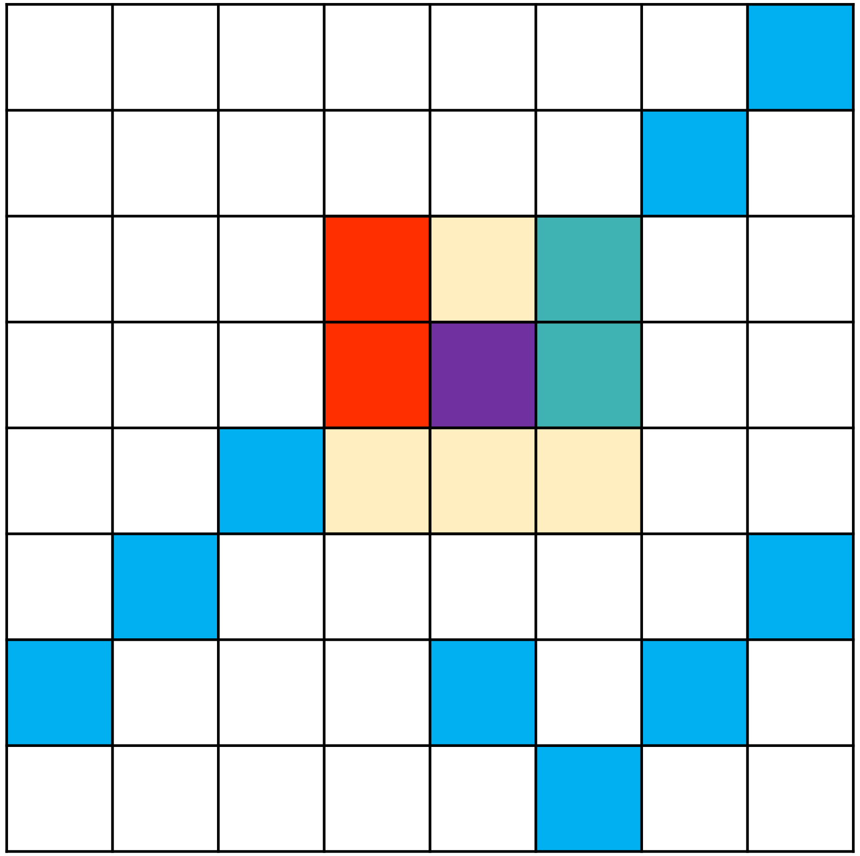
🟪 Now

🟫 Queue

Magnitude

Canny edge detection

- **Determine edge**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4)]



Magnitude

🟥 Strong Edge

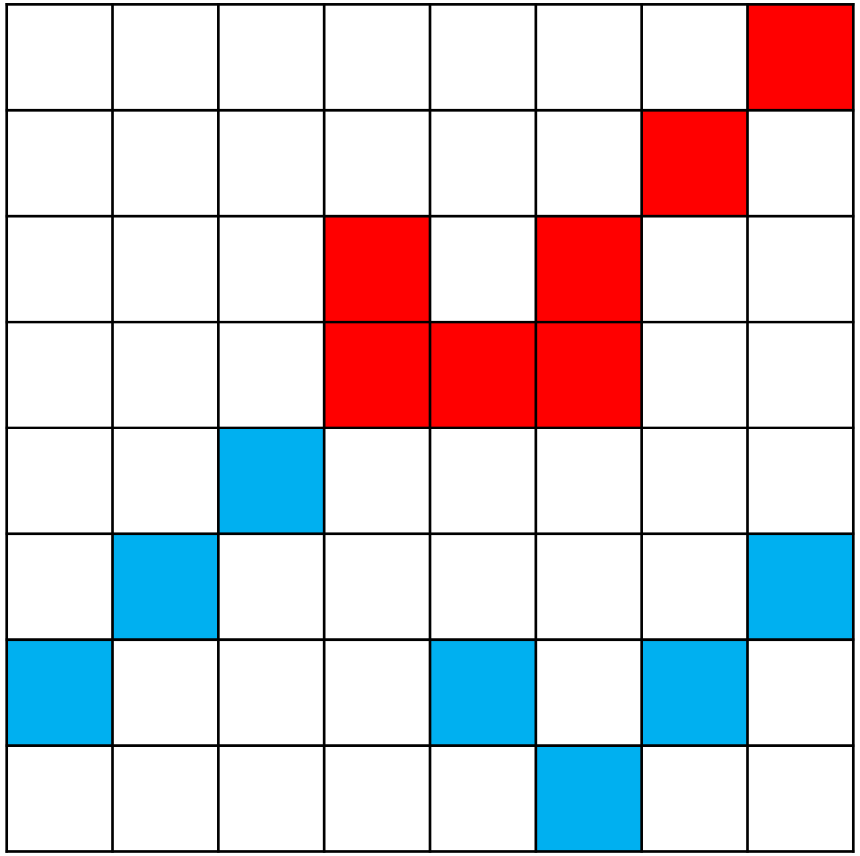
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: []



Magnitude

🟥 Strong Edge

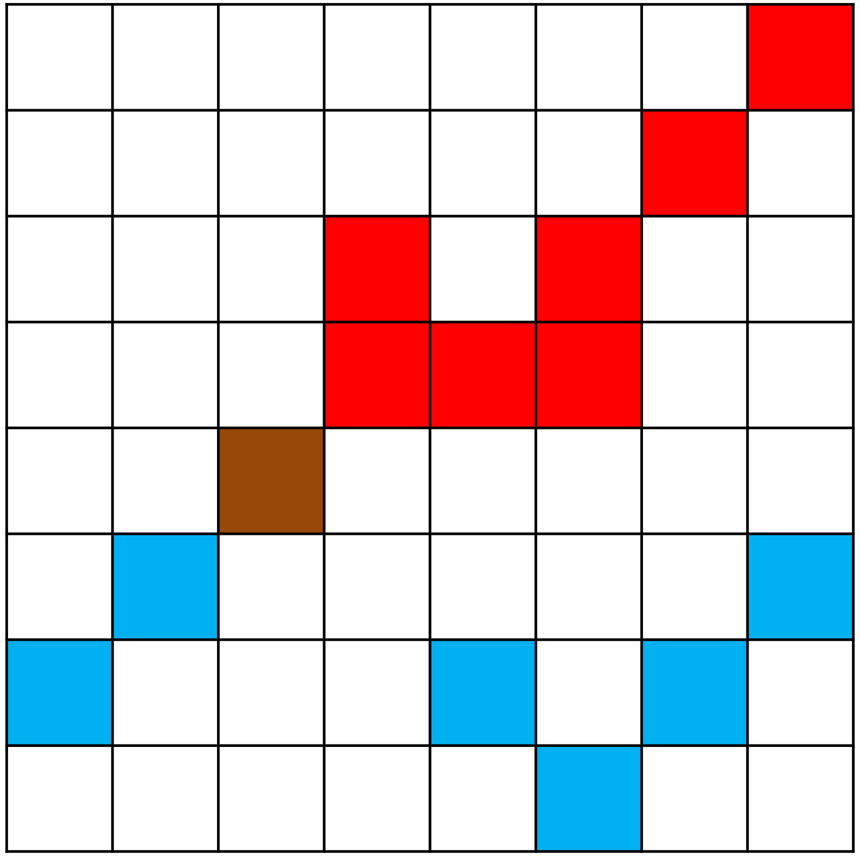
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [4, 2]

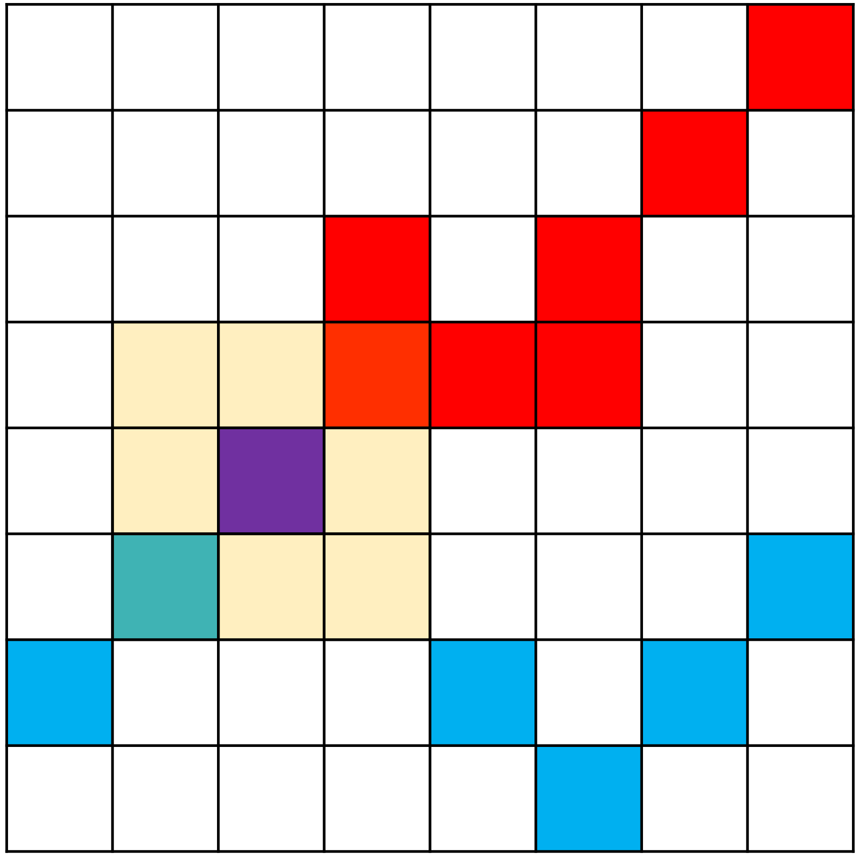


Magnitude

- 🟥 Strong Edge
- 🟦 Weak Edge
- 🟪 Now
- 🟫 Queue
- ⬛ Connect

Canny edge detection

- **Determine edge**
  - Connect: [4, 2]

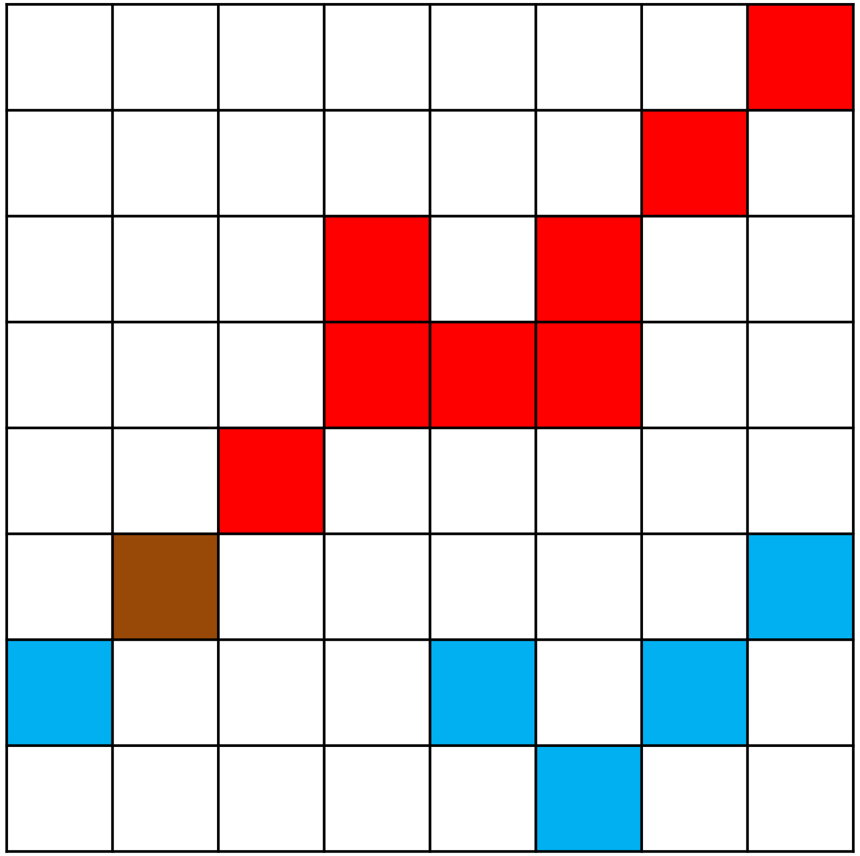


Magnitude

- 🟥 Strong Edge
- 🟦 Weak Edge
- 🟪 Now
- 🟫 Queue
- ⬛ Connect

Canny edge detection

- **Determine edge**
  - Connect: [(5, 1)]



Magnitude

🟥 Strong Edge

🟦 Weak Edge

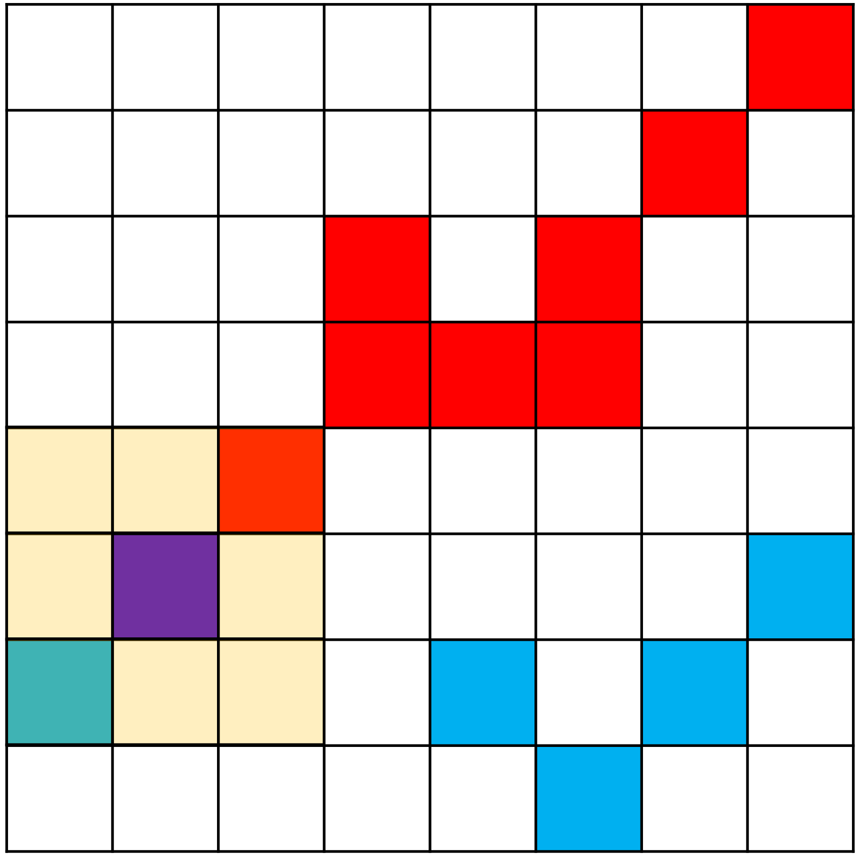
🟪 Now

🟫 Queue

⬛ Connect

Canny edge detection

- **Determine edge**
  - Connect: [(5, 1)]



Magnitude

🟥 Strong Edge

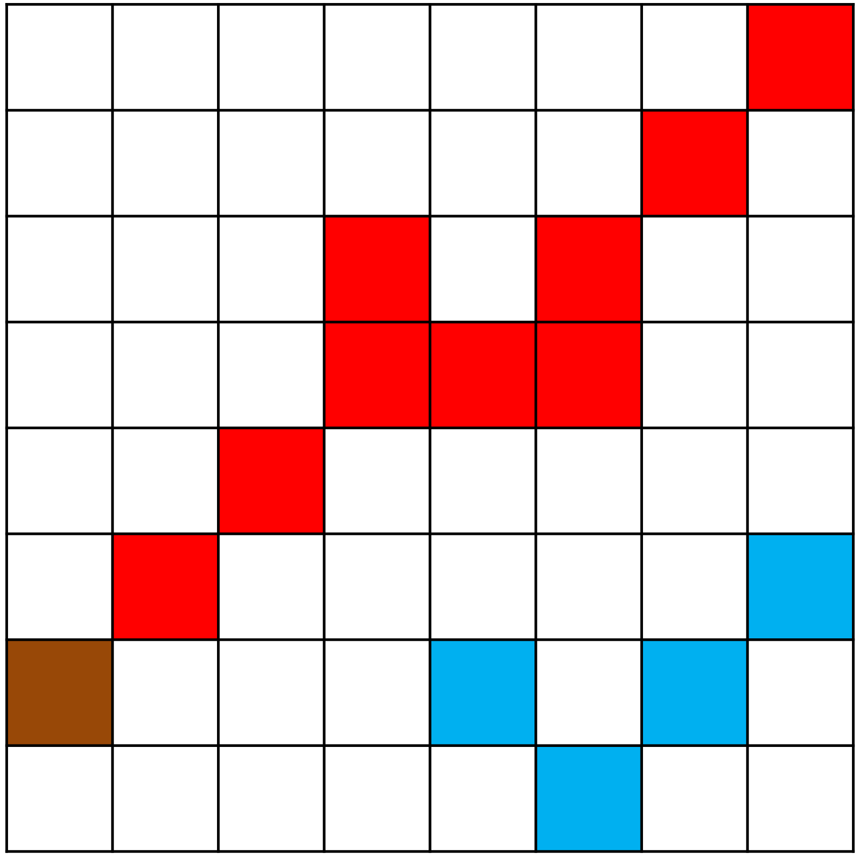
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [(6, 0)]



Magnitude

🟥 Strong Edge

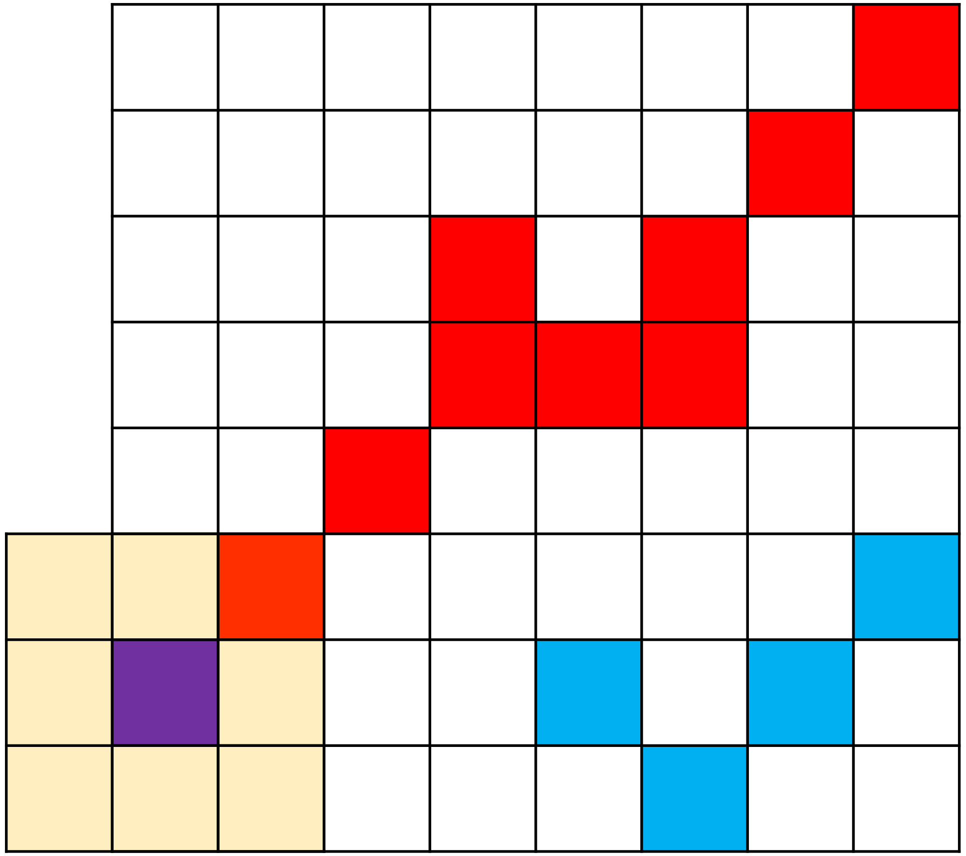
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [(6, 0)]



Magnitude

🟥 Strong Edge

🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: []

							🟥
						🟥	
			🟥		🟥		
			🟥	🟥	🟥		
		🟥					
	🟥						🟦
🟥				🟦		🟦	
					🟦		

Magnitude

🟥 Strong Edge

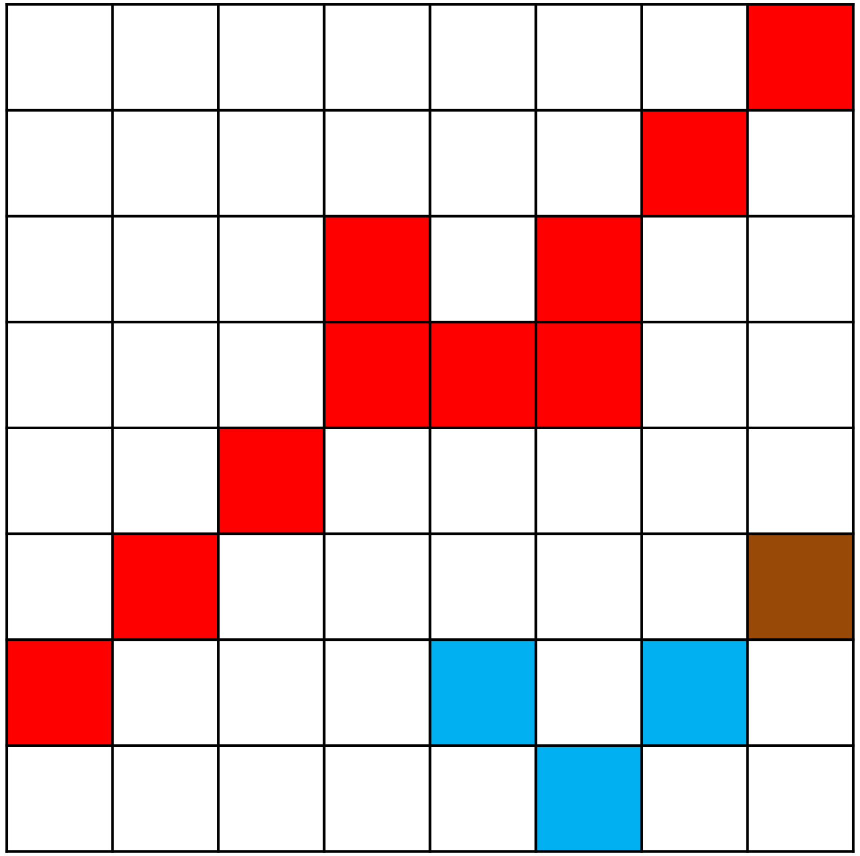
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [(5, 7)]

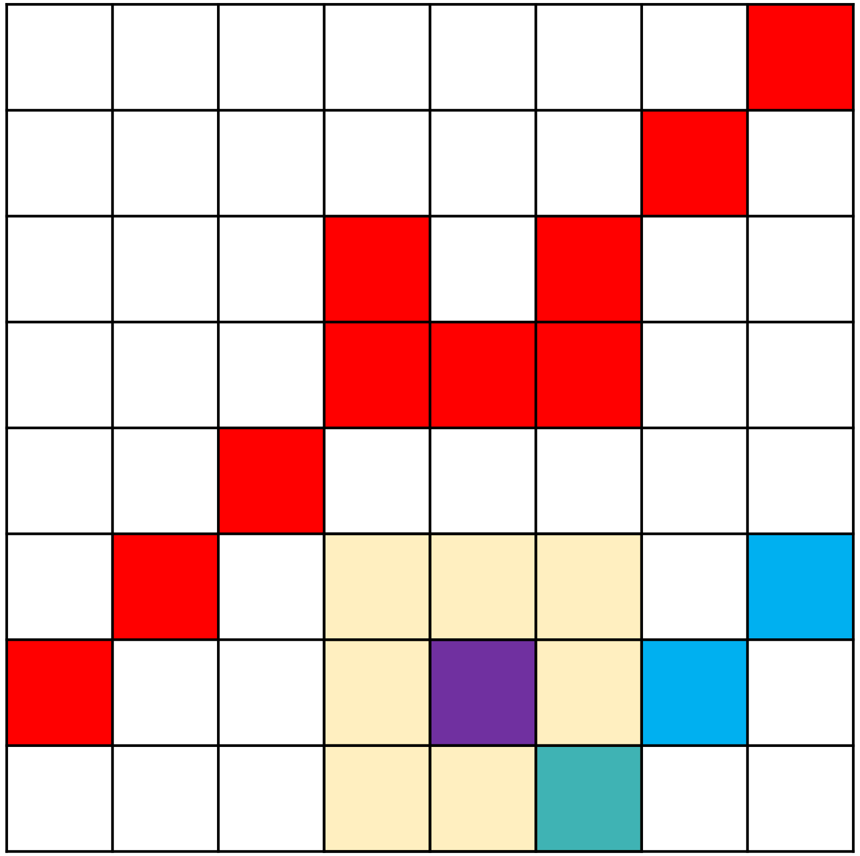


Magnitude

- 🟥 Strong Edge
- 🟦 Weak Edge
- 🟪 Now
- 🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: [(5, 7), (6, 6), (7, 5), (6, 4)]



Magnitude

🟥 Strong Edge

🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Determine edge**
  - Connect: []

							🟥
						🟥	
			🟥		🟥		
			🟥	🟥	🟥		
		🟥					
	🟥						
🟥							

Magnitude

🟥 Strong Edge

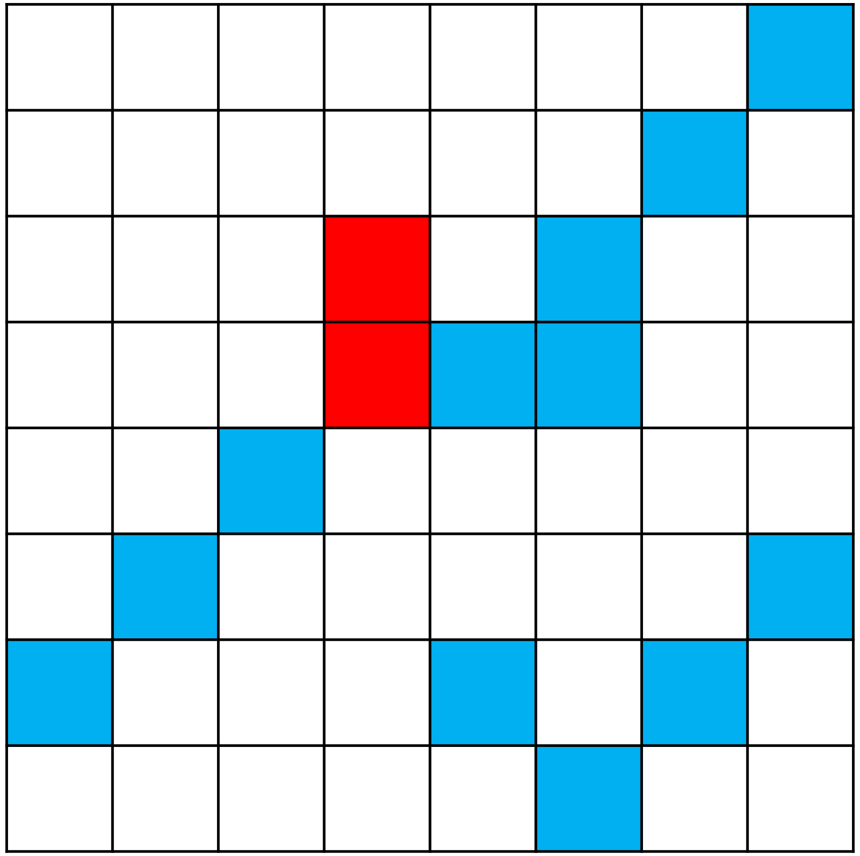
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: []



Magnitude

🟥 Strong Edge

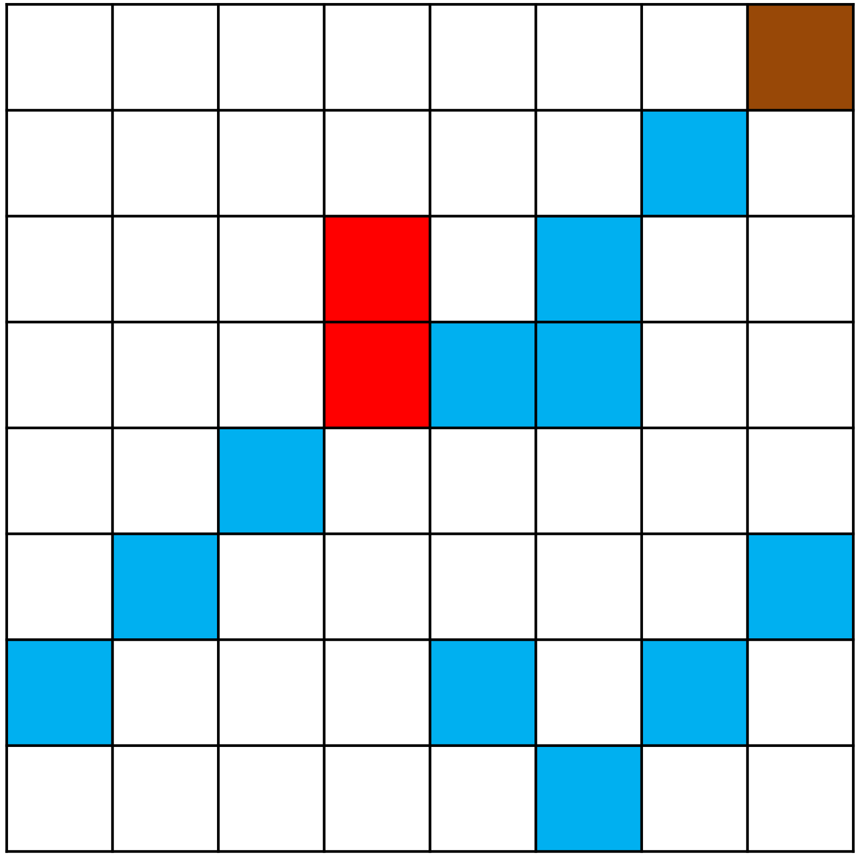
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7)]



Magnitude

🟥 Strong Edge

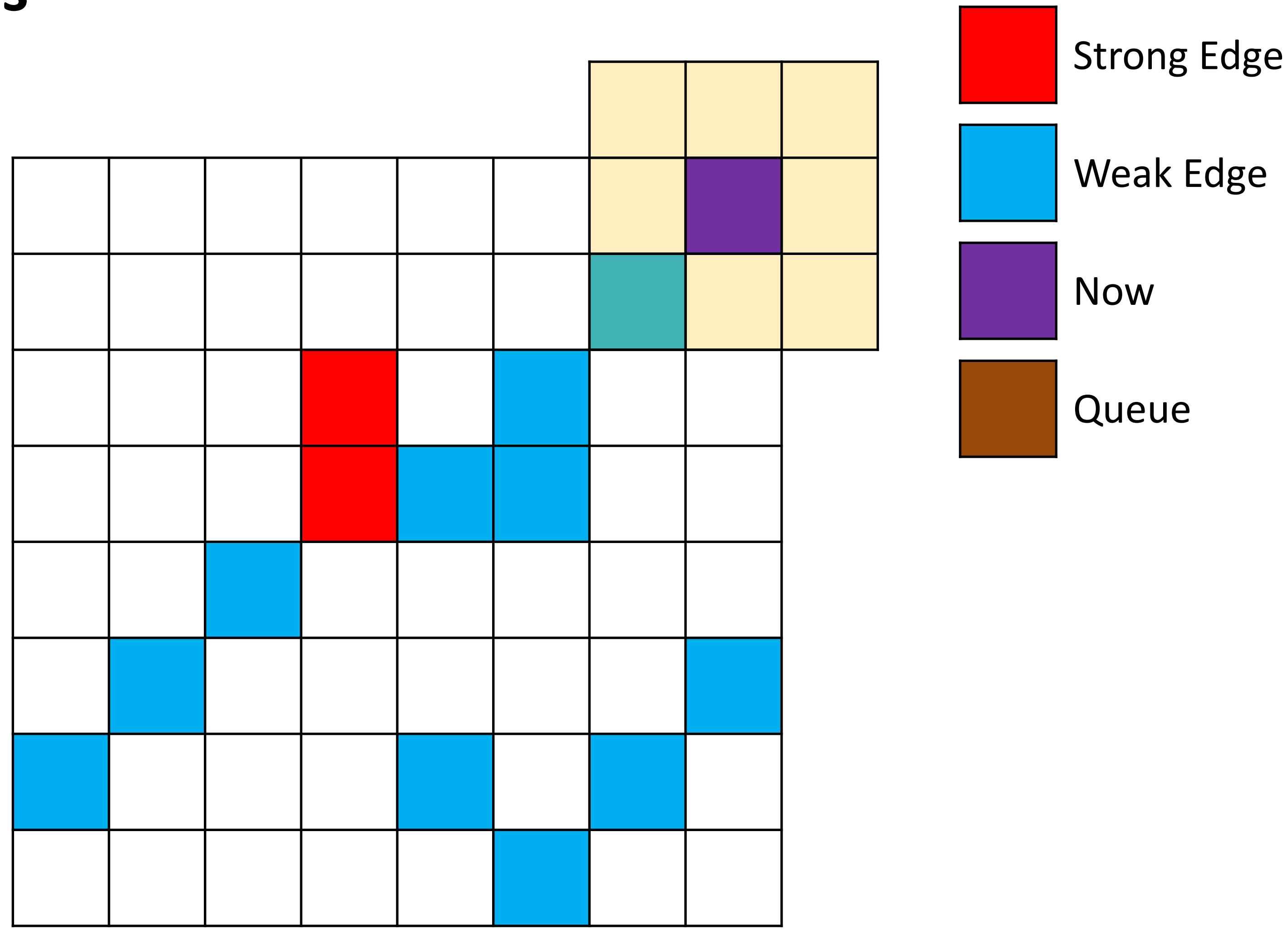
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

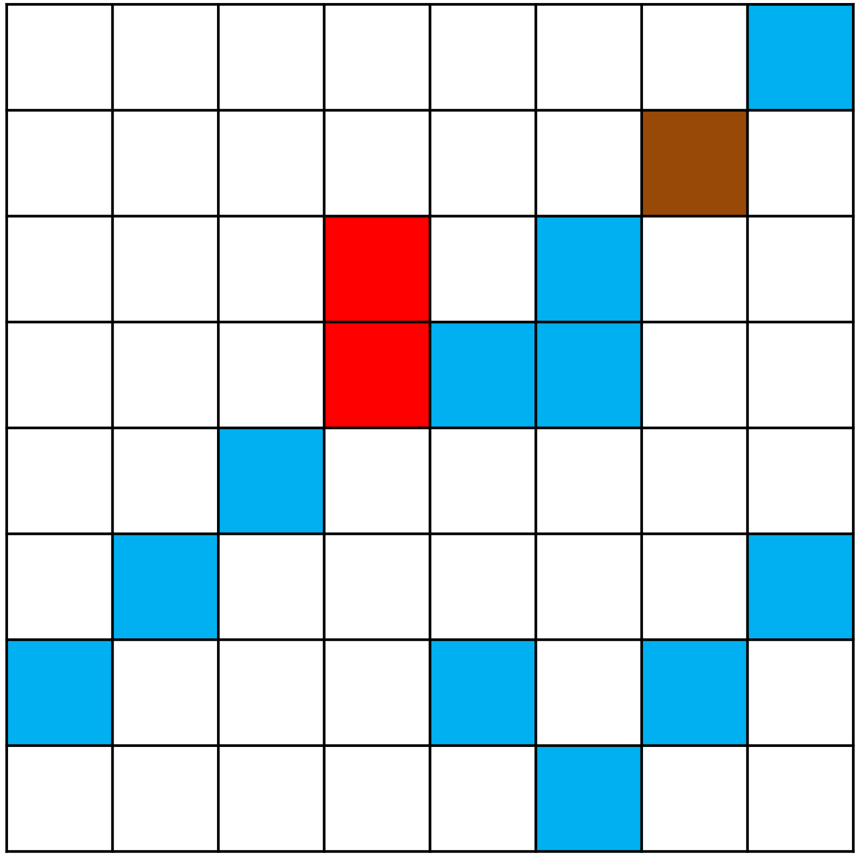
- **Connected components**
  - Connect: [(0, 7)]



Magnitude

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6)]



Magnitude

🟥 Strong Edge

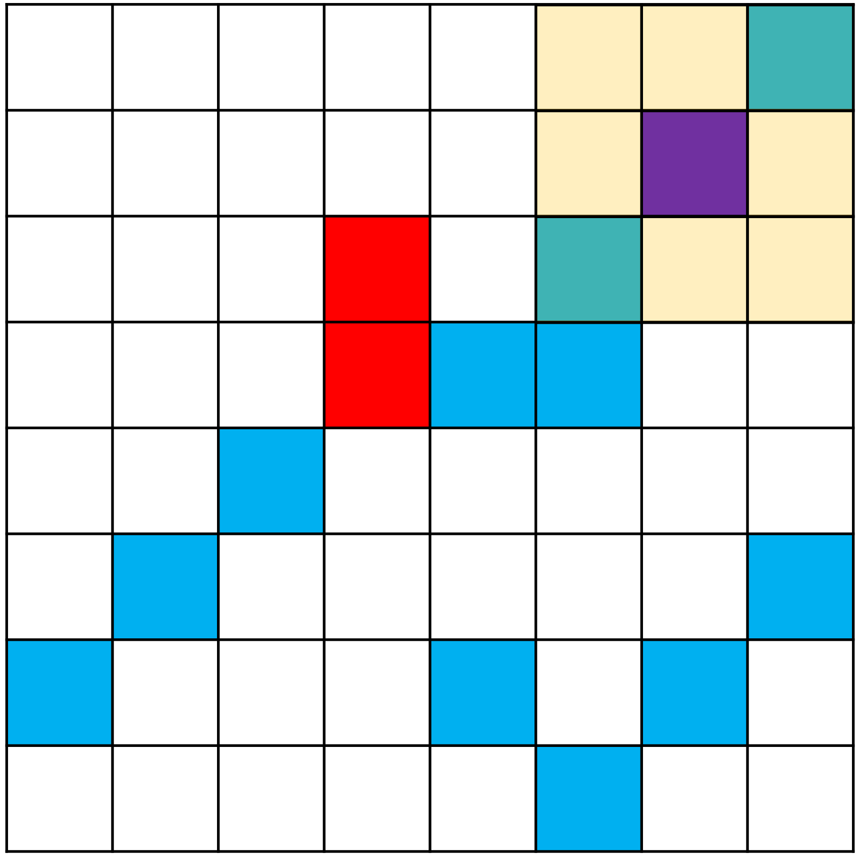
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6)]



Magnitude

🟥 Strong Edge

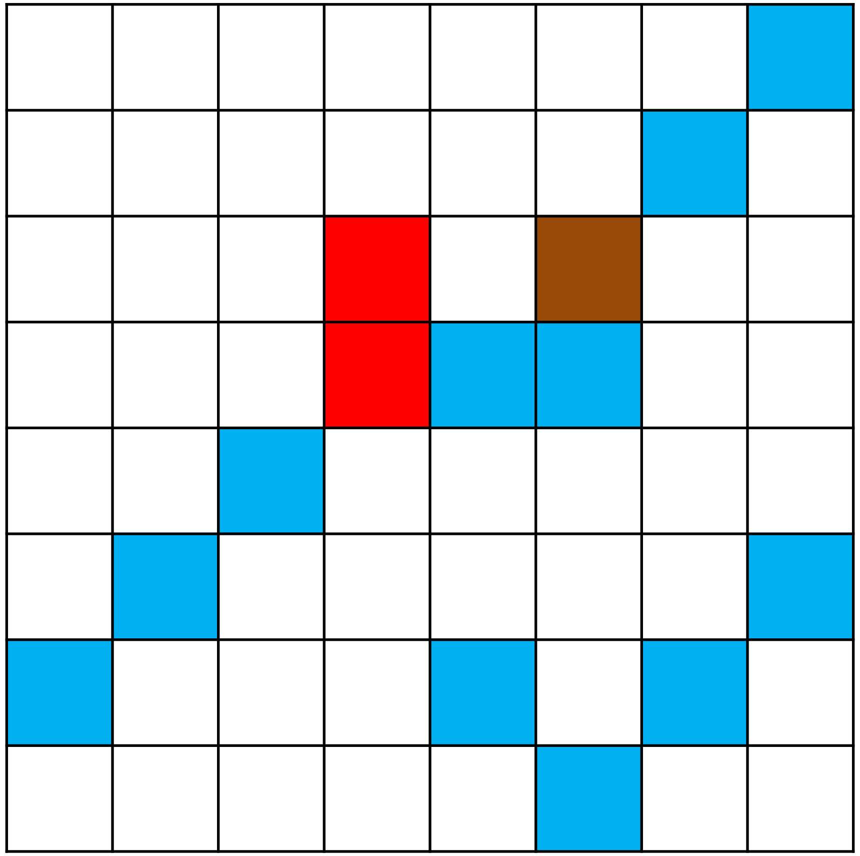
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5)]



Magnitude

🟥 Strong Edge

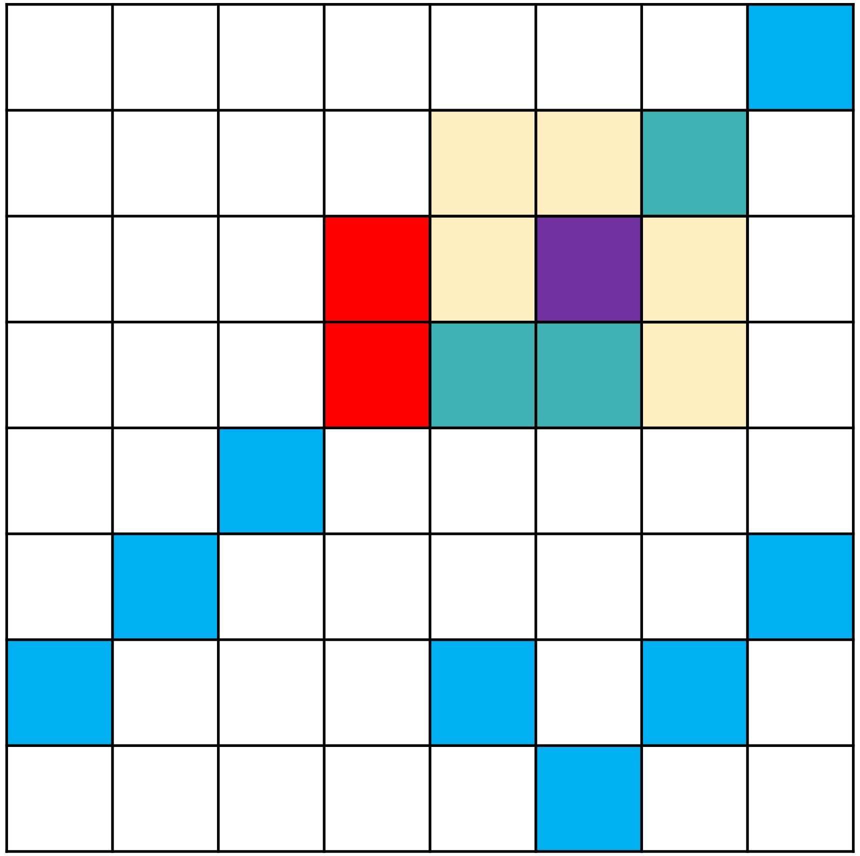
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5)]

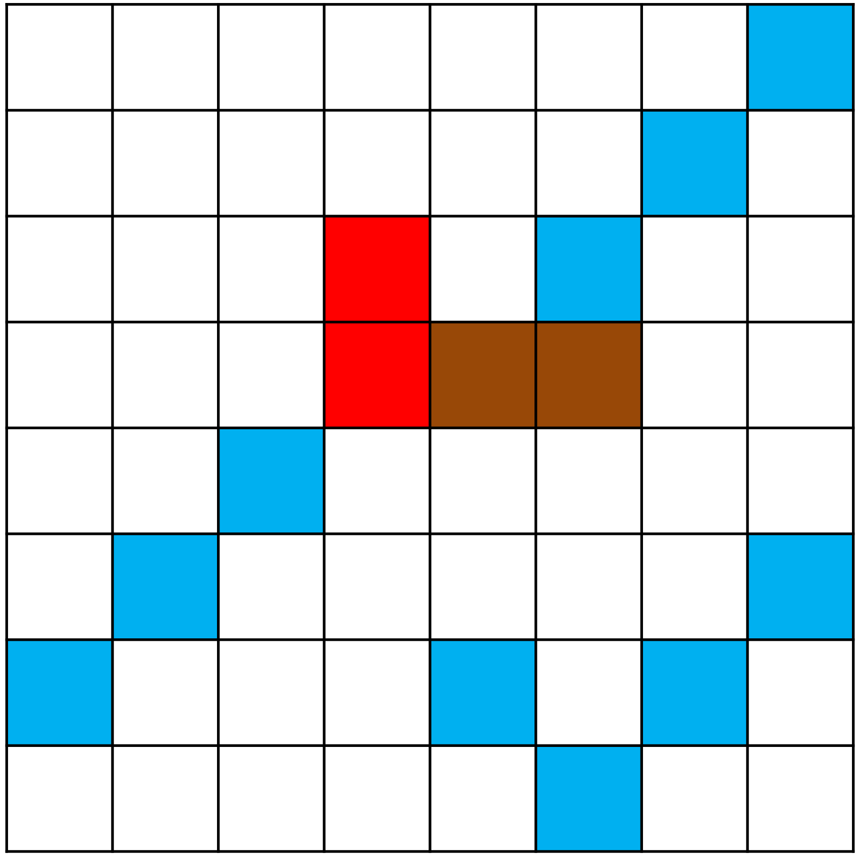


Magnitude

- 🟥 Strong Edge
- 🟦 Weak Edge
- 🟪 Now
- 🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4)]



Magnitude

🟥 Strong Edge

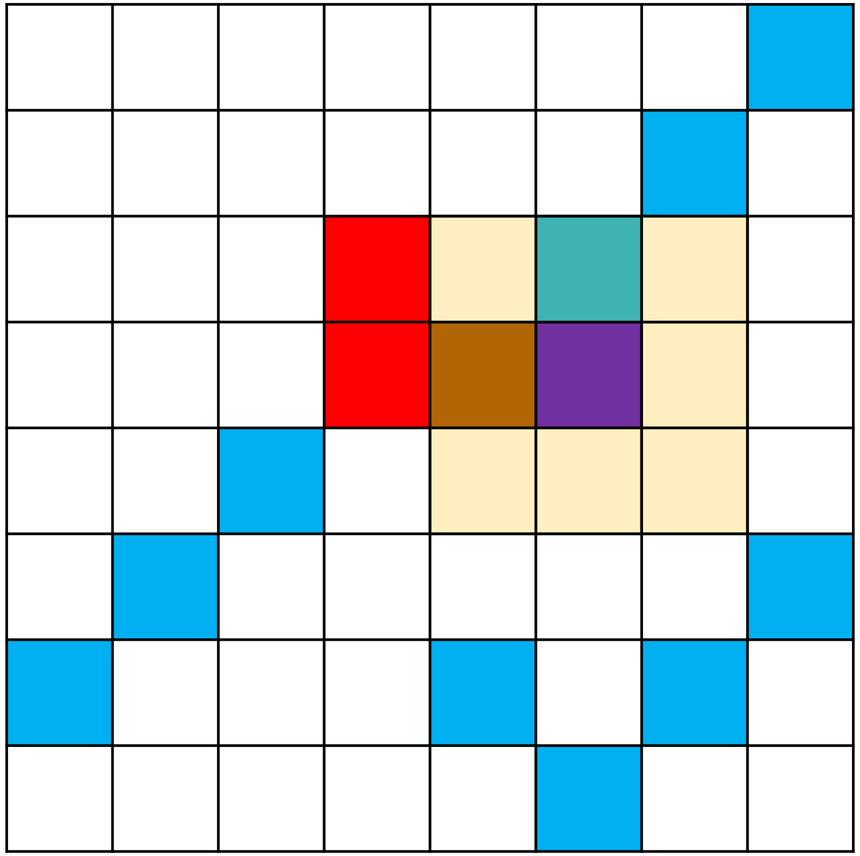
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4)]



Magnitude

🟥 Strong Edge

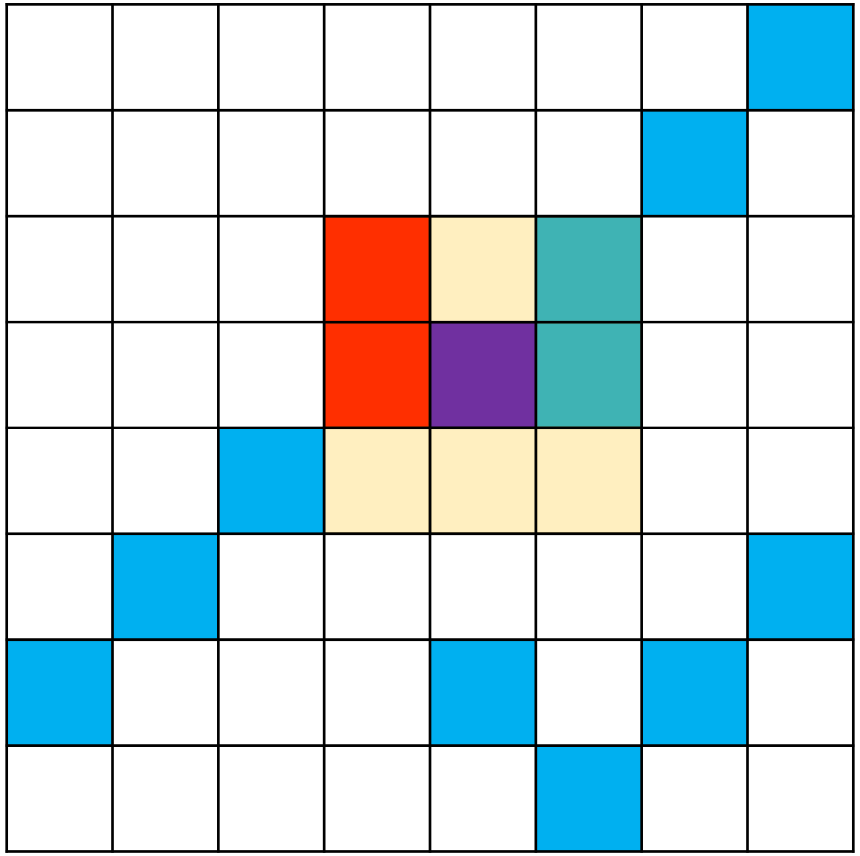
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4)]



Magnitude

🟥 Strong Edge

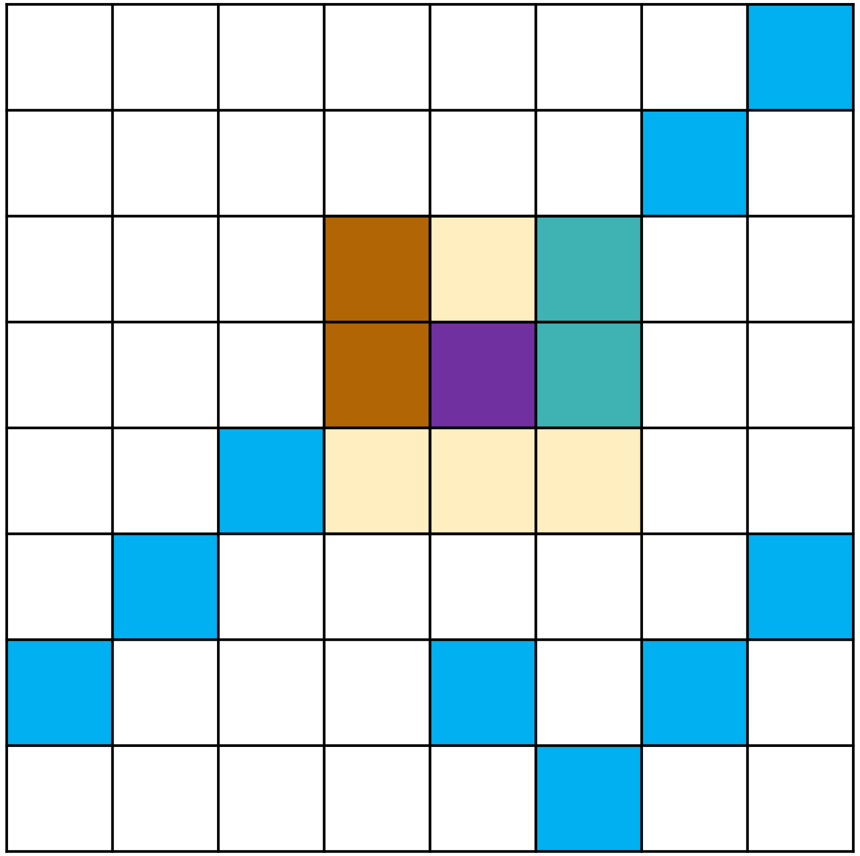
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4), (2, 3), (3, 3)]



Magnitude

🟥 Strong Edge

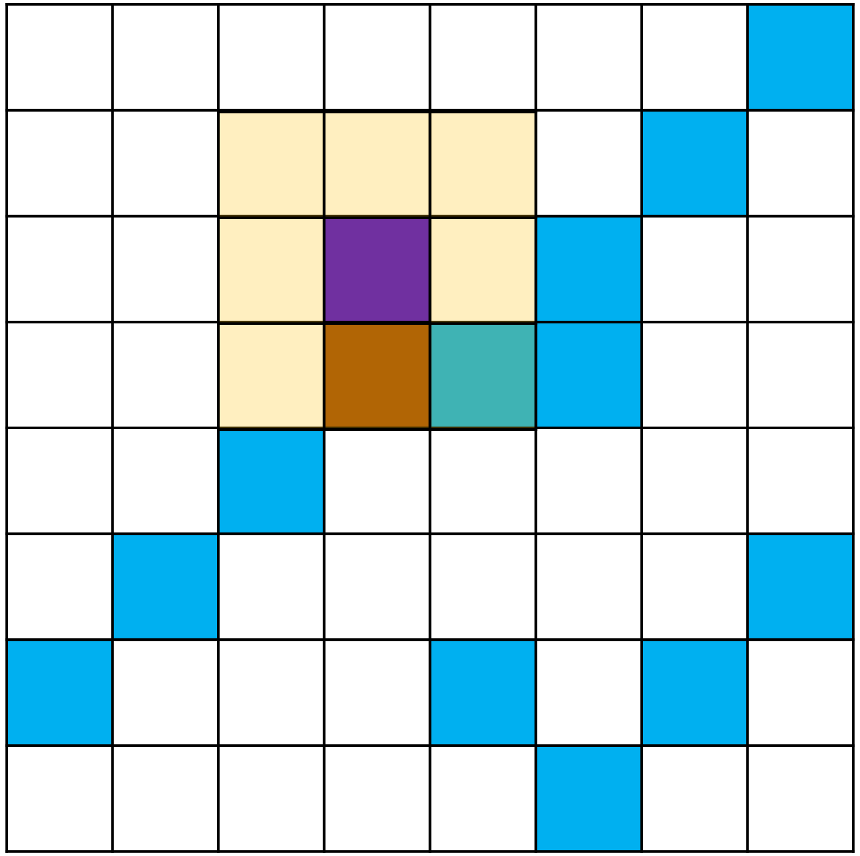
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4), (2, 3), (3, 3)]



Magnitude

🟥 Strong Edge

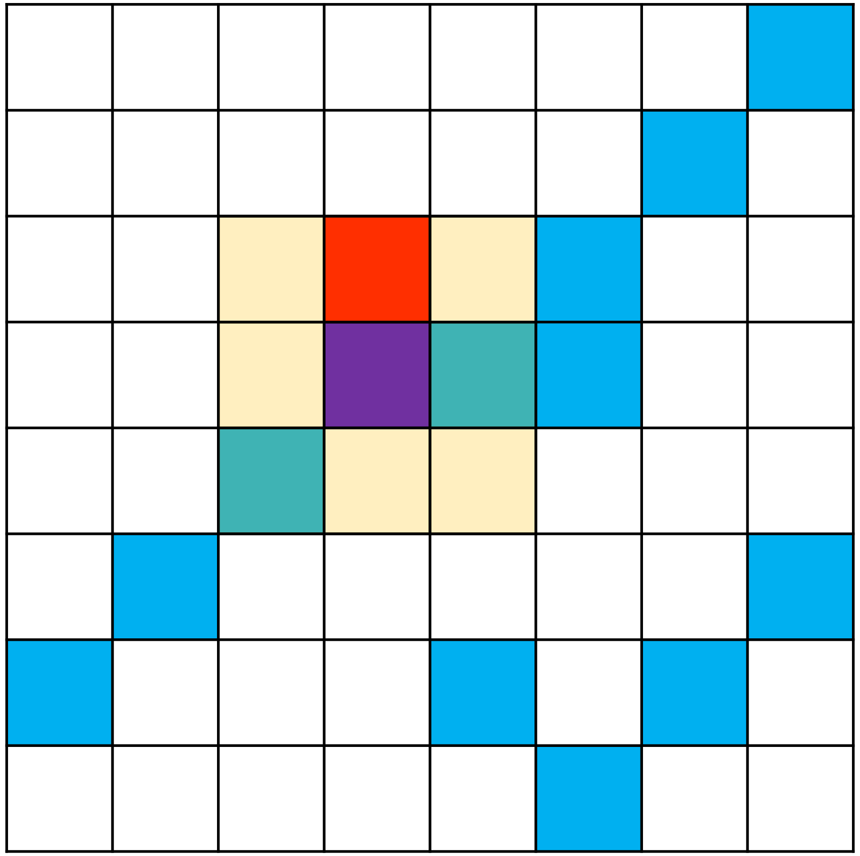
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4), (2, 3), (3, 3)]



Magnitude

🟥 Strong Edge

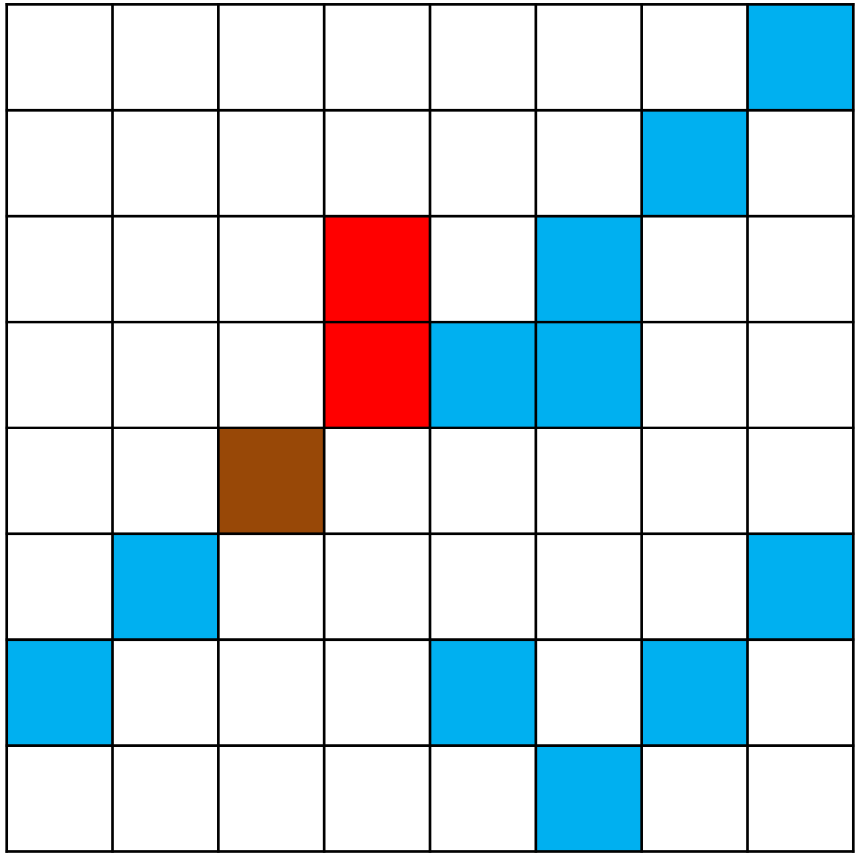
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4), (2, 3), (3, 3), (4, 2)]



Magnitude

🟥 Strong Edge

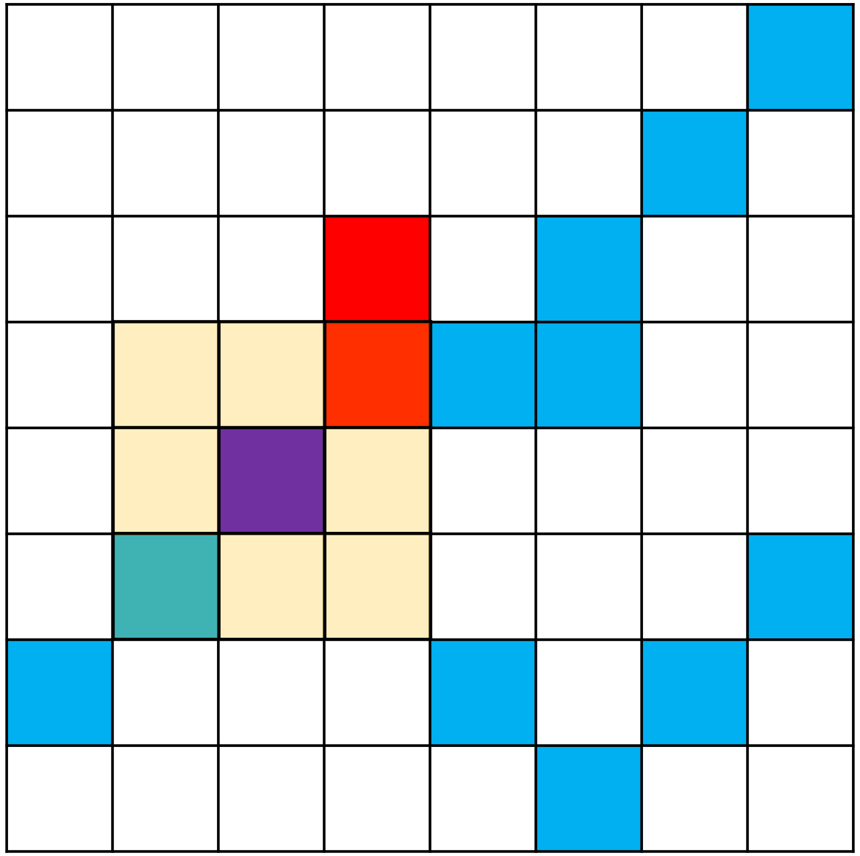
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4), (2, 3), (3, 3), (4, 2)]



Magnitude

🟥 Strong Edge

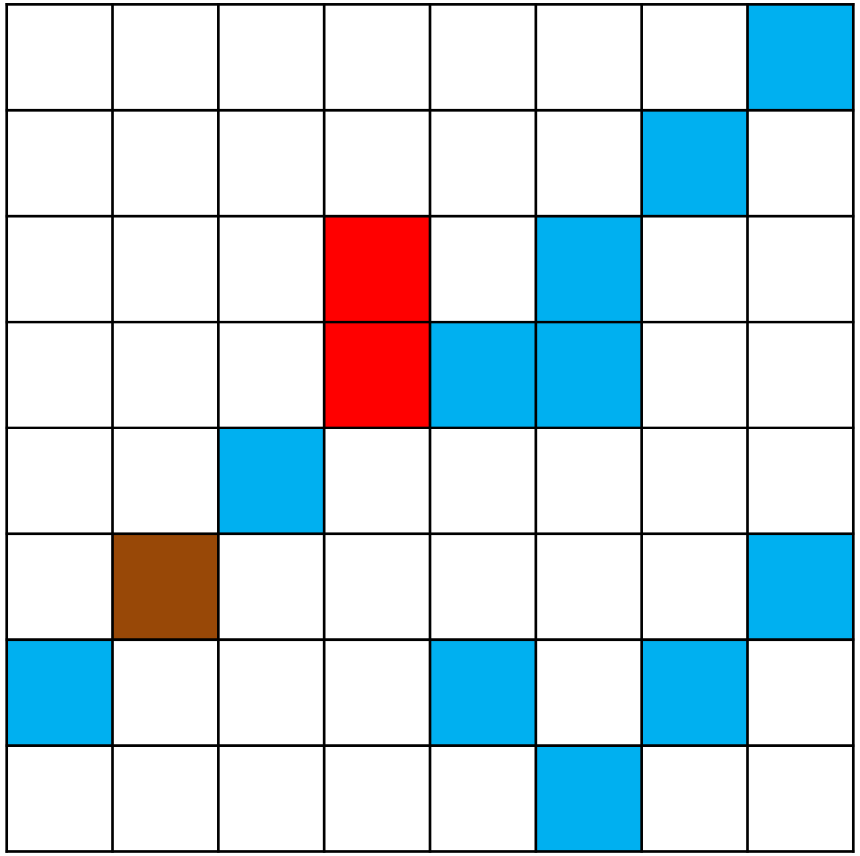
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4), (2, 3), (3, 3), (4, 2), (5, 1)]

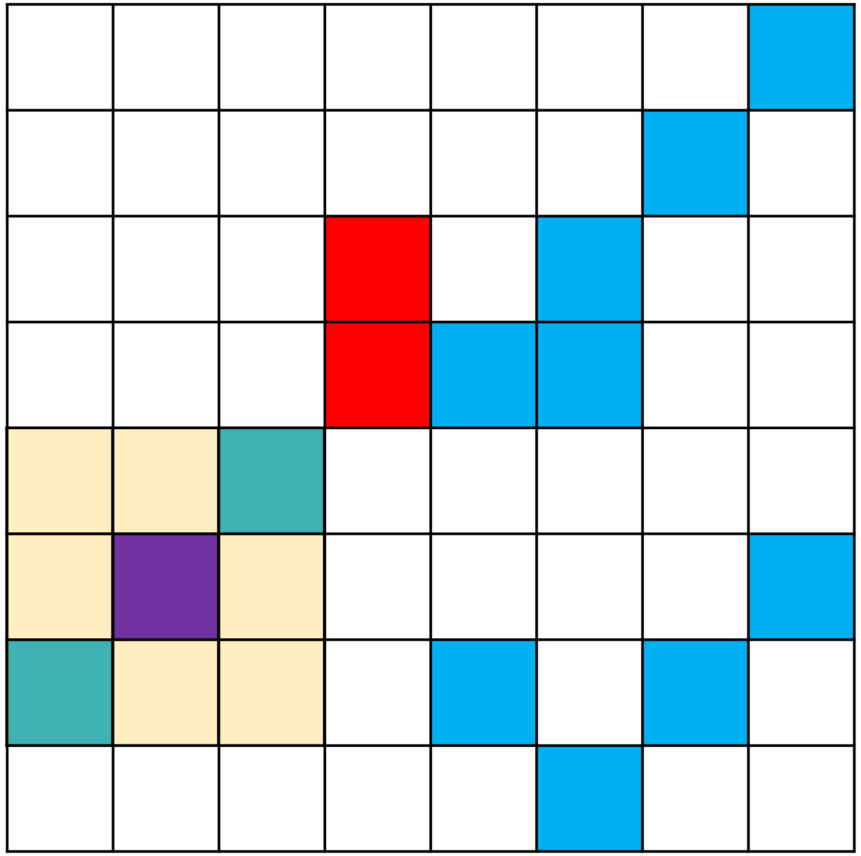


Magnitude

- 🟥 Strong Edge
- 🟦 Weak Edge
- 🟪 Now
- 🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4), (2, 3), (3, 3), (4, 2), (5, 1)]



Magnitude

🟥 Strong Edge

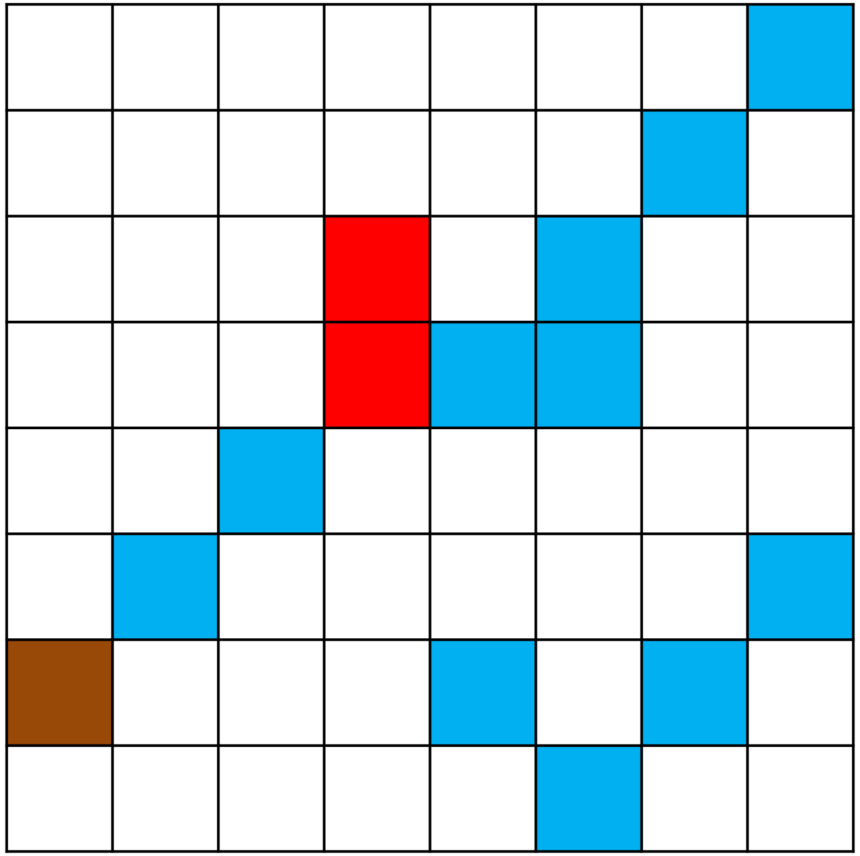
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4), (2, 3), (3, 3), (4, 2), (5, 1), (6, 0)]



Magnitude

🟥 Strong Edge

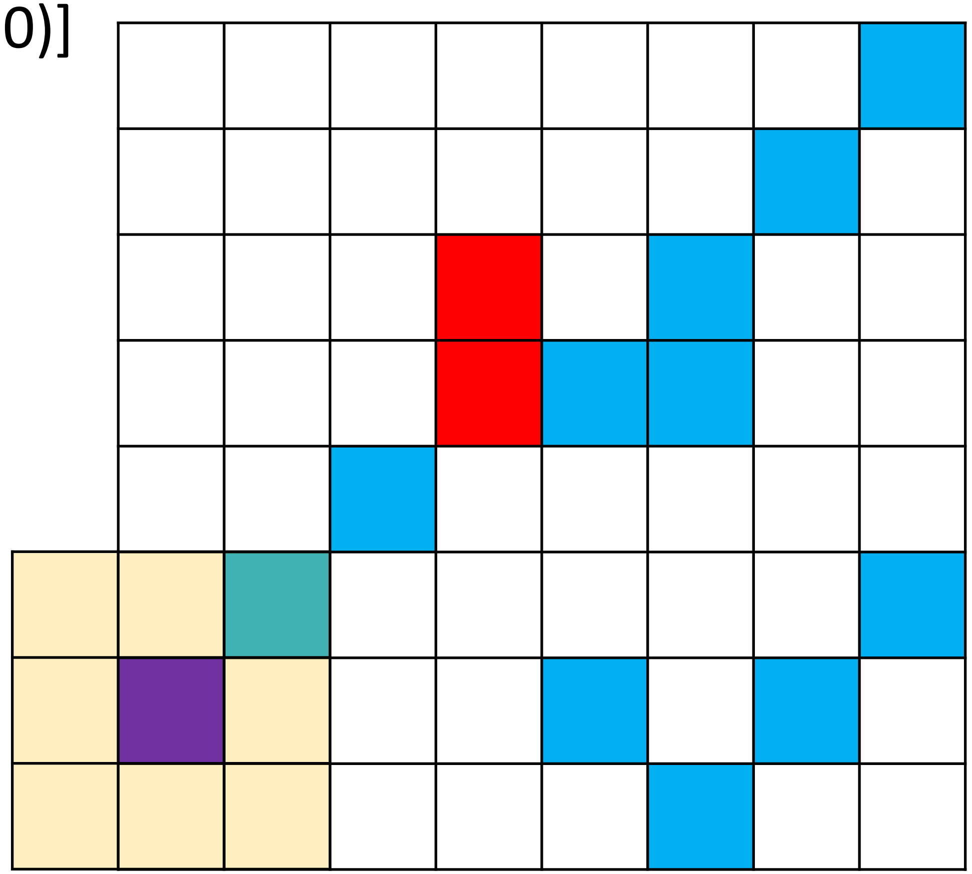
🟦 Weak Edge

🟪 Now

🟫 Queue

Canny edge detection

- **Connected components**
  - Connect: [(0, 7), (1, 6), (2, 5), (3, 5), (3, 4), (2, 3), (3, 3), (4, 2), (5, 1), (6, 0)]



Magnitude

- 🟥 Strong Edge
- 🟦 Weak Edge
- 🟪 Now
- 🟫 Queue

Canny edge detection

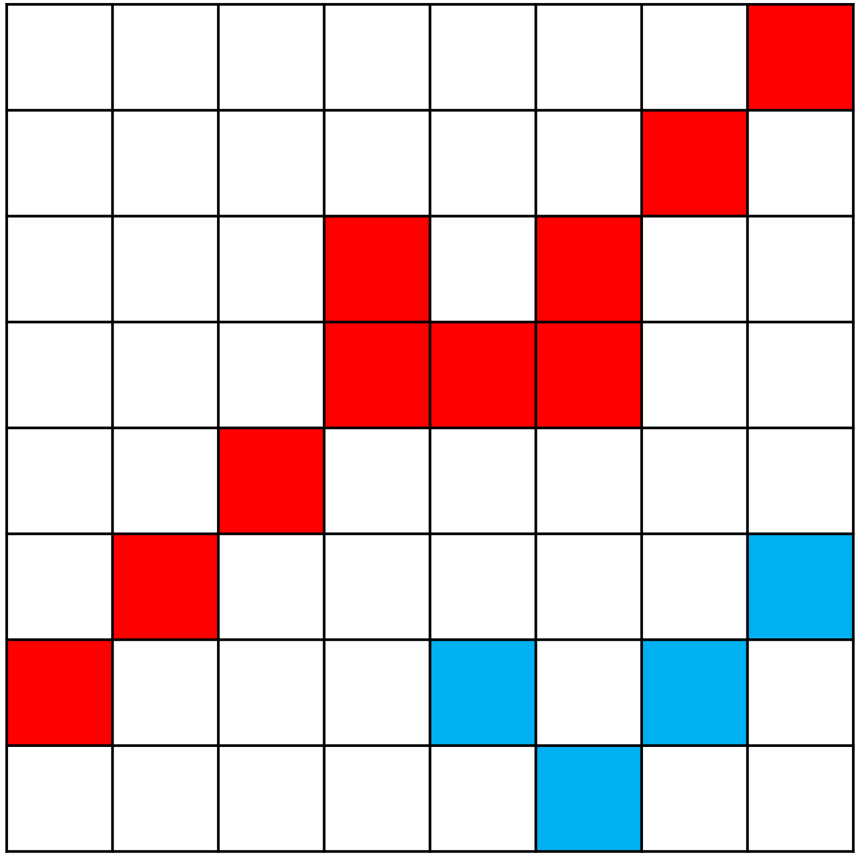
- **Connected components**
  - Connect: []

🟥 Strong Edge

🟦 Weak Edge

🟪 Now

🟫 Queue



Magnitude

과제 canny_edge_detection.py

- 함수 설명
  - DoG_x, DoG_y = get_DoG_filter(fsize=5, sigma=1)
  - gradient_y = cv2.filter2D(image, -1, DoG_y)
  - gradient_x = cv2.filter2D(image, -1, DoG_x)
  - magnitude = calculate_magnitude(gradient_x, gradient_y)
  - nms_result = non_maximum_suppression(gradient_x, gradient_y, magnitude, n=5)
  - thresholding_result = double_thresholding(nms_result, high_threshold=10, low_threshold=4)
  - canny_edge_result = determine_edge(thresholding_result)

과제 canny_edge_detection.py

- **nms_result = non_maximum_suppression(gradient_x, gradient_y, magnitude, n=5):**
  - gradient_x: dog filtering을 통해 나온 gradient_x
  - gradient_y: dog filtering을 통해 나온 gradient_y
  - magnitude: dog filtering을 통해 나온 magnitude
  - n: non maximum suppression할 영역크기
  - nms_result: non maximum suppression 결과

Canny edge detection

- **thresholding_result = double_thresholding(nms_result, high_threshold=10, low_threshold=4):**
  - nms_result: non maximum suppression 결과
  - high_threshold, low_threshold: edge를 구분할 threshold 2개
    - high_threshold > low_threshold
  - thresholding_result: strong edge, weak edge, not edge로 구분한 2차원 행렬
    - Strong edge인 경우(nms_result[y, x] > high_threshold): 해당 좌표의 픽셀 값 255
    - Weak edge인 경우(low_threshold ≤ nms_result[y, x] ≤ high_threshold): 해당 좌표의 픽셀 값 128
    - Not edge인 경우 (nms_result[y, x] < low_threshold): 해당 좌표의 픽셀 값 0

Canny edge detection

- **canny_edge_result: determine_edge(thresholding_result):**
  - thresholding_result : double thresholding 결과
  - canny_edge_result: weak edge(128)를 high edge(255) 또는 not edge(0)로 바꾼 행렬

weak_edge = np.where(thresholding_result == 128)
→ 2 크기의 튜플: 조건에 해당하는 y좌표, x좌표

좌표 조회: [(-1, -1), (-1, 0), (-1, 1), (0, 1), (1, 1), (1, 0), (1, -1), (0, -1)]

과제

- **보고서**
  - 내용
    - 학과, 학번, 이름
    - 구현 코드: 구현한 코드에 대한 간단한 설명
    - 이미지: 5x5 DoG filter(Sigma=1)입힌 magnitude, 5x5 non-maximum-suppression 후 이미지, double thresholding 후 이미지, 8-neighborhood로 determine edge 후 이미지 총 4장
    - 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]20xxxxxx_이름_x주차_과제.pdf

과제

- **과제 요약**
  - 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]20XXXXXX_이름_X주차_과제.zip 으로 제출
      - .py 파일
      - .pdf 보고서 파일
  - 제출 기한
    - 2024년 5월 9일 23시 59분까지

Q & A