



# LECTURE 03 IMAGE HISTOGRAM AND SPATIAL FILTERING

Punnarai Siricharoen, Ph.D.

## OBJECTIVES

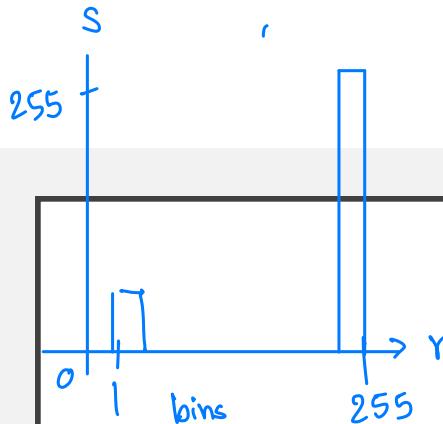
- To understand how to describe an image using histogram and histogram equalization techniques
- To understand filtering techniques, e.g., image smoothing or sharpening

# CONTENT

- Histogram
- Spatial Filtering

# HISTOGRAM





## HISTOGRAM PROCESSING



- A histogram is a graphical representation of a data distribution that uses bars to display the frequency or count of data points falling within specific intervals, known as "bins".
- It provides a visual way to understand the distribution of data and identify patterns, trends, and outliers.
- Histogram manipulation used for image enhancement, compression and segmentation.
- Provide useful statistics.

Intensity 5  
L = 5  
0 - 4

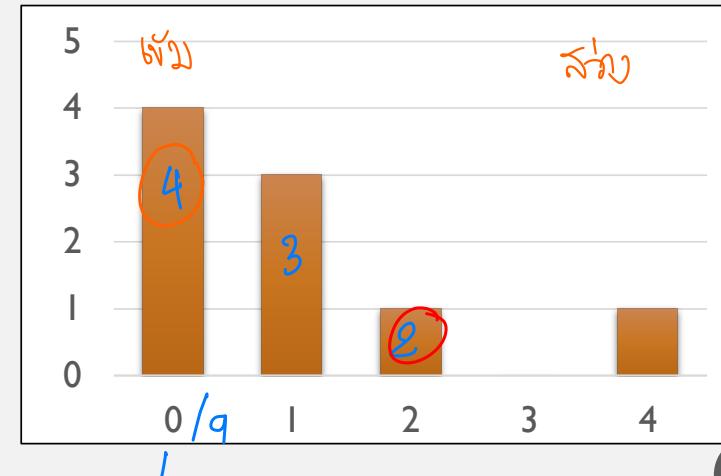
0	0	1
0	1	0
1	2	4

A 3x3 image

9 px



Occurrence frequency ( $n_k$ )



0 4 px  
Intensity level ( $r_k$ )

# HISTOGRAM PROCESSING

$$L = 5$$

$$0 - 4$$

- The histogram of a digital image with gray levels in the range  $[0, L - 1]$  is a discrete function:

$$h(r_k) = n_k$$

where  $r_k$  is the kth gray level and  $n_k$  is the number of pixels in the image having gray level  $r_k$ .

- A normalized histogram:

คำนวณทั้งหมดแล้วได้

$$p(r_k) = n_k / n_{\text{px}}$$

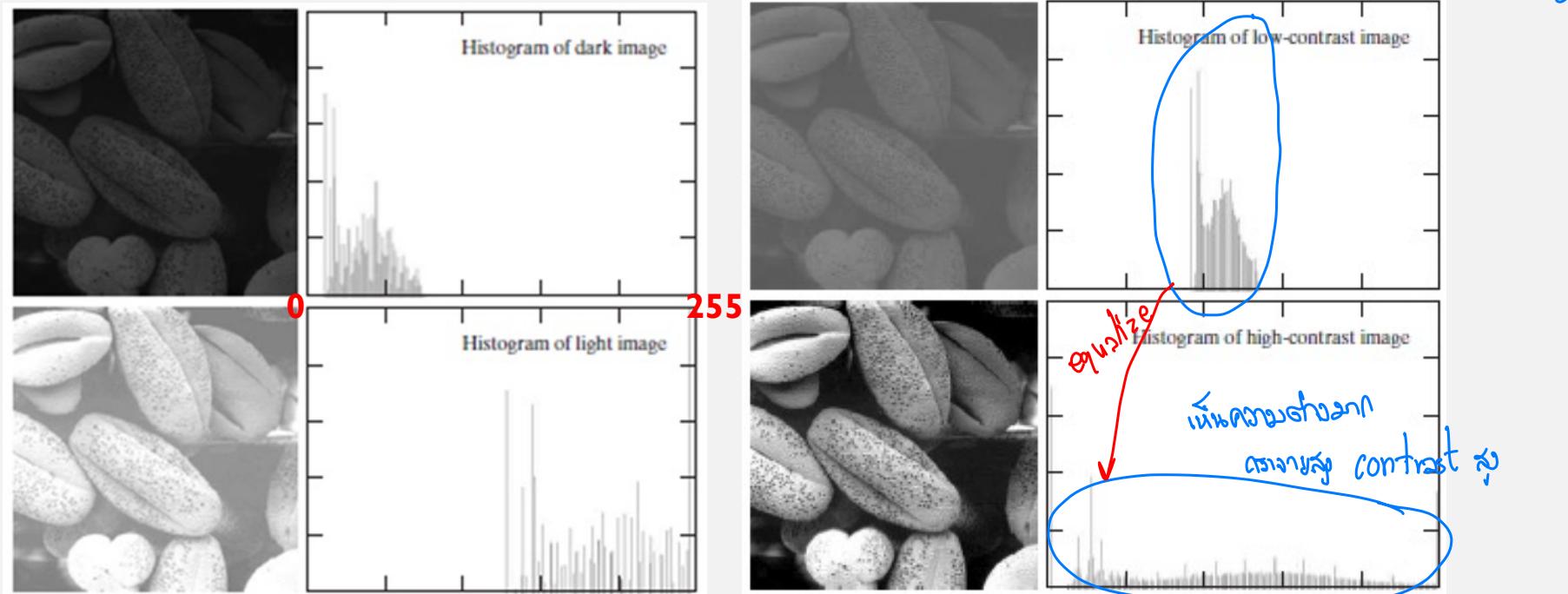
- where  $n$  is the total number of pixels.

- $p(r_k)$  is an estimate of the probability of occurrence of gray level  $r_k$ ; sum of all  $p(r_k) = 1$ .

# HISTOGRAM PROCESSING



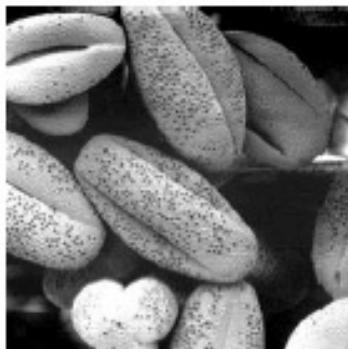
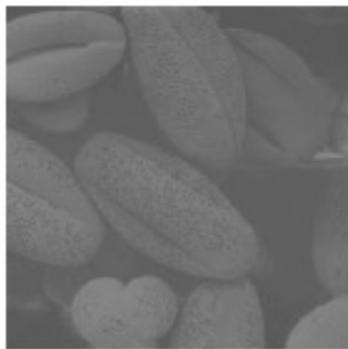
- Intensity levels in the range  $[0, 255]$  (8-bit image)



- Minimum, maximum, and average intensity values.

## HISTOGRAM EQUALIZATION

គ្រឿង contrast ក្នុង histogram ក្រោមាយព័ត៌មានក្នុង<sup>ពេនិជ្ជកម្មសំខាន់ៗ</sup>



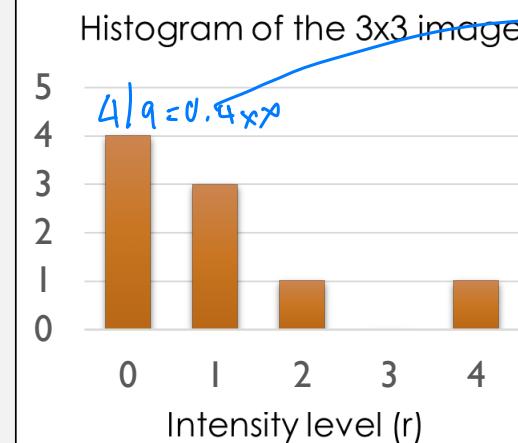
- ▶ Brighten some dark values and darken some light values?
- ▶ Histogram equalization: to find an intensity mapping function  $T$  resulting histogram is flat.
$$s = T(r)$$
- ▶ Trick: find probability density function (PDF) to compute cumulative distribution function (CDF)

# HISTOGRAM EQUALIZATION

0	0	1
0	1	0
1	2	4

3x3 image

Histogram



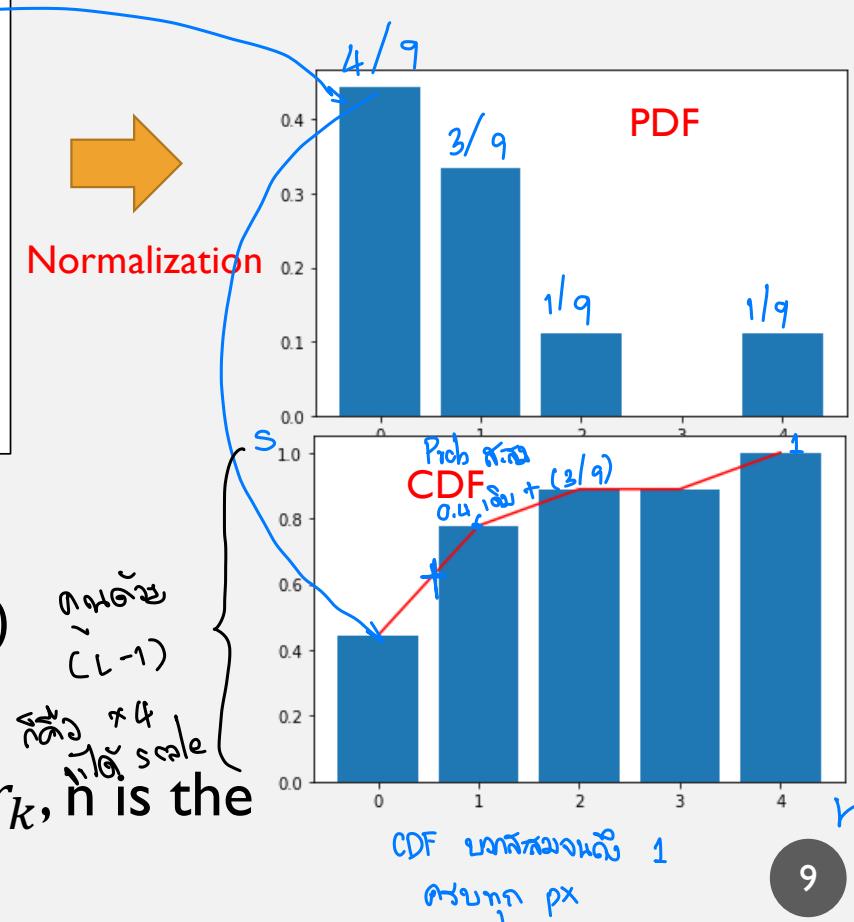
Cumulative distribution function: CDF

$$c(r_k) = \frac{1}{n} \sum_{j=0}^k h(r_j) = c(r_{k-1}) + p(r_k)$$

รวม hist ต่อไป

where  $h$  is histogram value at gray level  $r_k$ ,  $n$  is the total number of pixels.

T

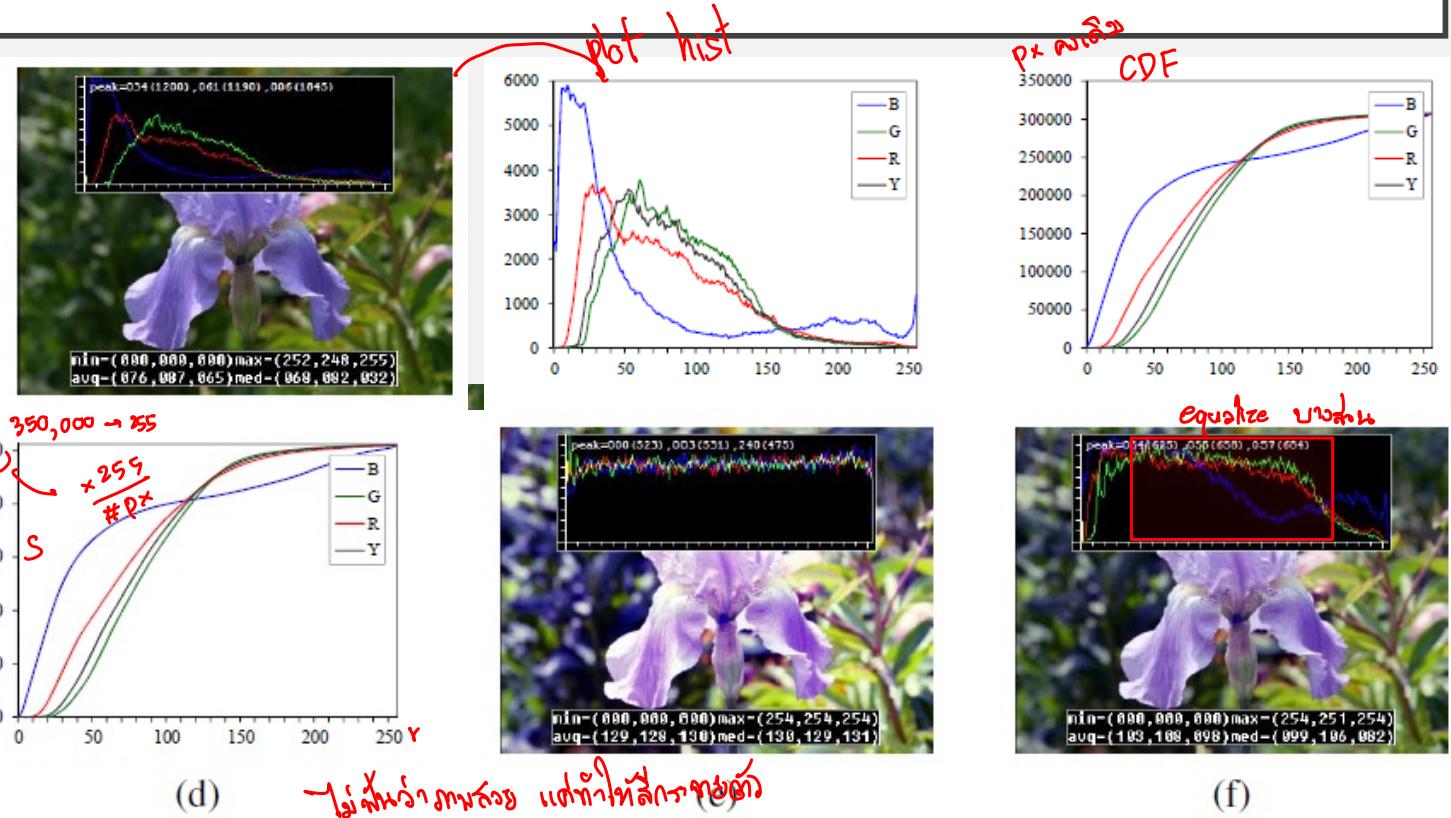
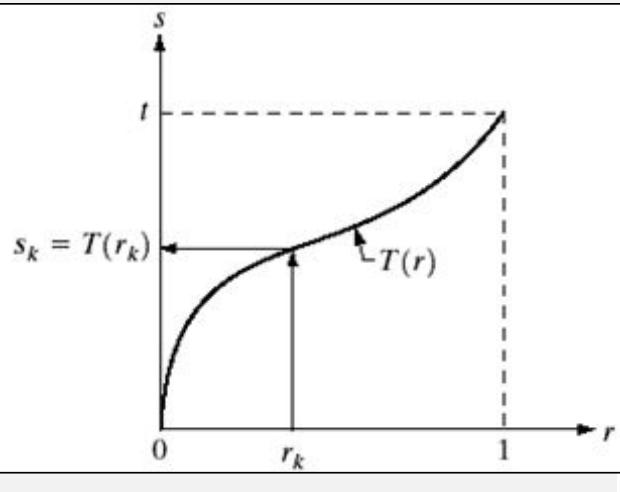


หมายความว่า จ่ายตัง น้ำหนัก ใหม่ - ใหม่

# HISTOGRAM EQUALIZATION

Considering normalized r  
to the interval [0, 1]:

$$s = T(r)$$



**Figure 3.7** Histogram analysis and equalization: (a) original image (b) color channel and intensity (luminance) histograms; (c) cumulative distribution functions; (d) equalization (transfer) functions; (e) full histogram equalization; (f) partial histogram equalization.

## HISTOGRAM EQUALIZATION

Enhance its contrast by stretching out its histogram - Histogram Equalization

1. Our image  $L$  different gray levels:  $0, 1, 2, \dots, L - 1$
2. Gray level  $i$  occurs  $n_i$  times in the image.
3. We change gray level  $i$  to

$$\left( \frac{n_0 + n_1 + \dots + n_i}{N} \right) (L - 1)$$

$N$  is the total number of  
image pixel

4. Rounded to the nearest integer.

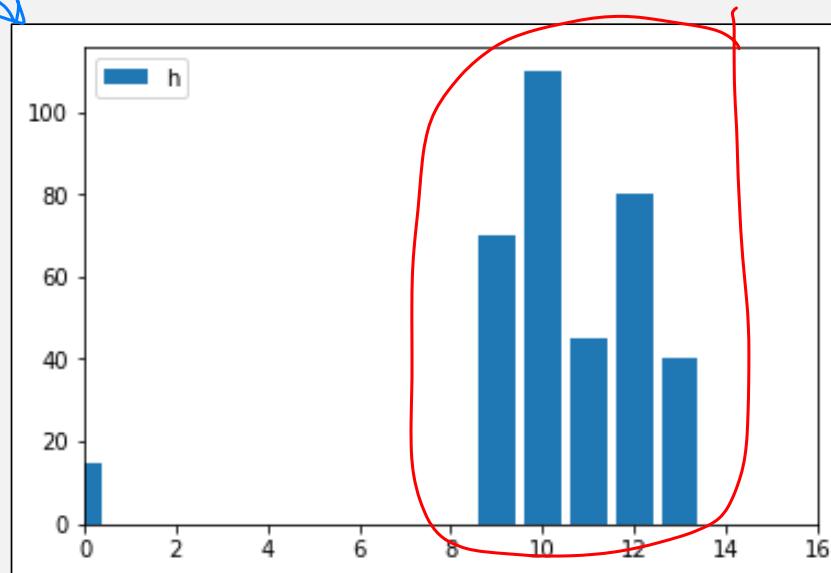
# HISTOGRAM EQUALIZATION

- Histogram Equalization

$L = 16 [0, 15]$

```
bar(x, height, *, align='center', **kwargs)
bar(x, height, width, *, align='center', **kwargs)
bar(x, height, width, bottom, *, align='center', **kwargs)
```

```
gray_level = np.arange(0,16,1)
n_i = np.array([15, 0, 0, 0, 0, 0, 0, 0, 0, 70, 110, 45, 80, 40, 0, 0])
plt.bar(gray_level,n_i)
```



# HISTOGRAM EQUALIZATION

- Histogram Equalization

①  $L = 16 \rightarrow [0, 15]$  list សម្រាប់

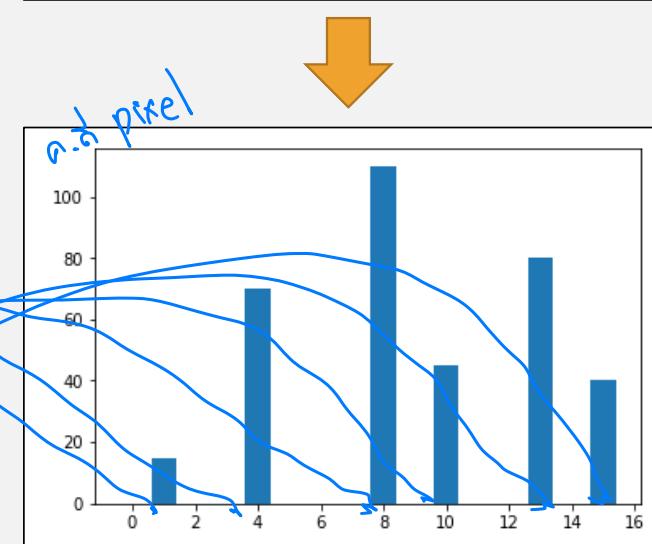
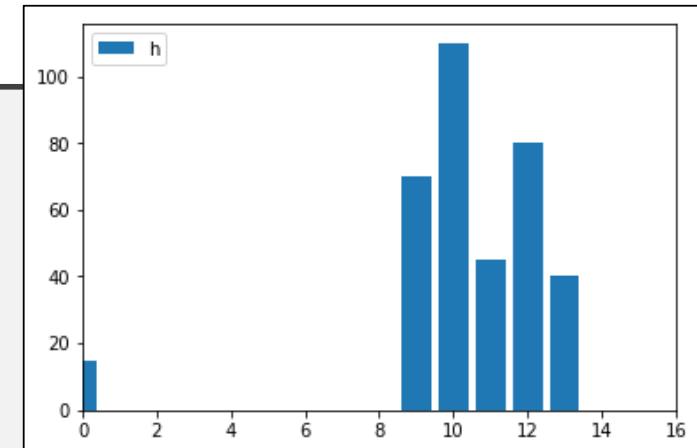
②  $m$  hist នៃលំដាប់ សម្រាប់

$$\left( \frac{L-1}{n} \right) = 15/360 = 1/24$$

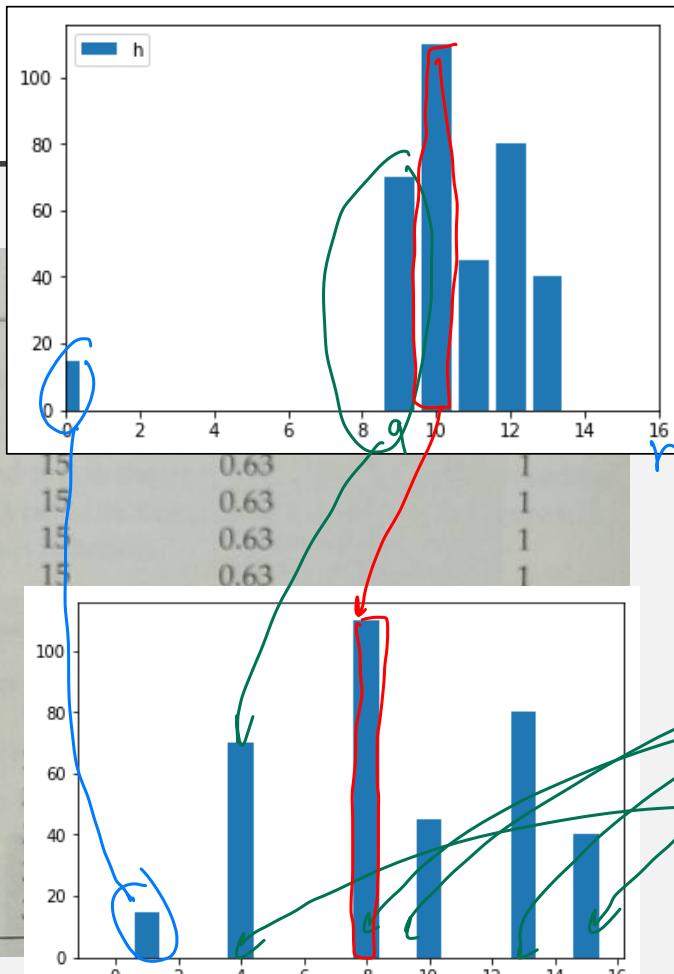
scale នៃលំដាប់ 0-15  
q.s. px

Gray level $i$	$\Sigma n_i$	$(1/24)\Sigma n_i$	Rounded value
0	15	0.63	1
1	0	0.63	1
2	0	0.63	1
3	0	0.63	1
4	0	0.63	1
5	0	0.63	1
6	0	0.63	1
7	0	0.63	1
8	0	0.63	1
9	70	3.65	4
10	110	8.13	8
11	45	10	10
12	80	13.33	13
13	40	15	15
14	0	15	15
15	0	15	15

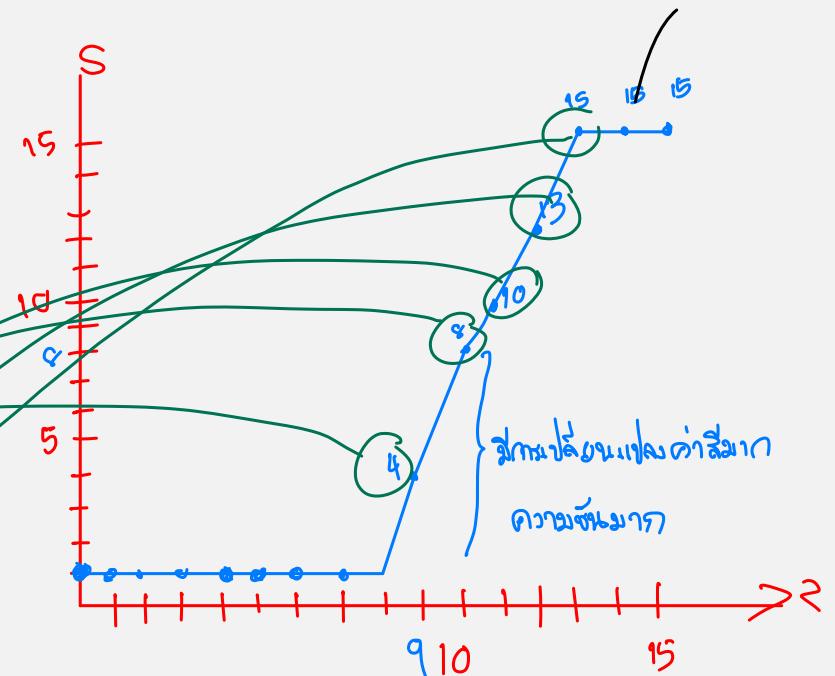
q.s. px នៃលំដាប់



Gray level $i$	$n_i$
0	15
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	70
10	110
11	45
12	80
13	40
14	0
15	0



เรขาคณิต plot transformation  
function ( $T$ ) ณ cumulative



จุดที่เปลี่ยนแปลงค่าสีมาก

## HISTOGRAM EQUALIZATION

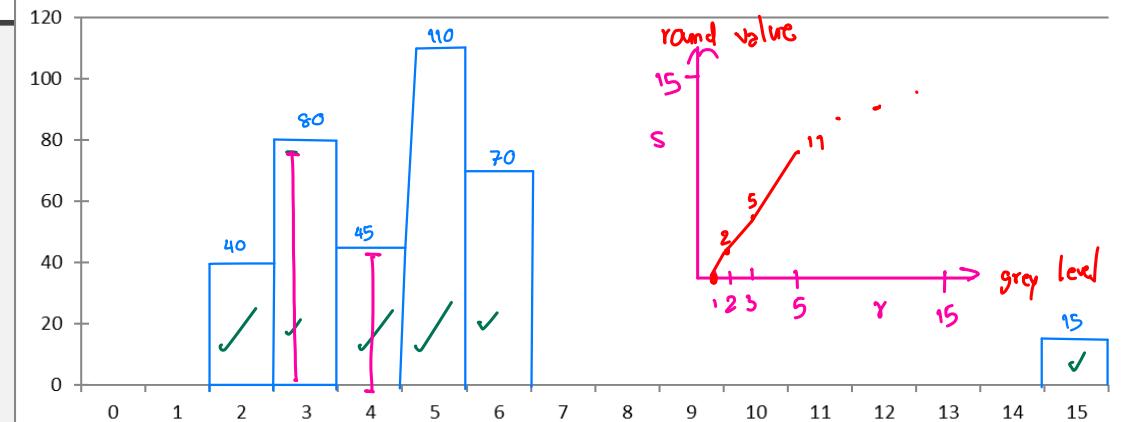
- Histogram Equalization
- EX#1: the number of pixels at each of the gray levels 0-15 (gray value only)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	40	80	45	110	70	0	0	0	0	0	0	0	0	15

- Perform a histogram equalization and draw the resulting histogram.

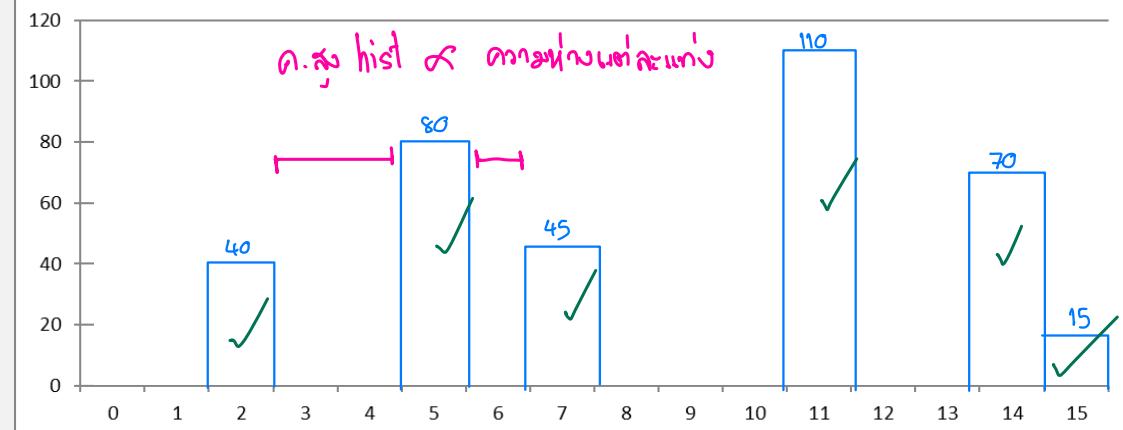
# HISTOGRAM EQUALIZATION

Original Histogram



gray level	$n_i$	$sum(n_i)$	$(1/360 * 15) * sum(n_i)$	round value
0	40	40	1.66	2
1	80	120	5	5
2	165	285	6.875	7
3	275	560	11.458	11
4	345	905	14.375	14
5	360	1265	14.375	14
6	70	1335	14.375	14
7	0	1335	0	0
8	0	1335	0	0
9	0	1335	0	0
10	0	1335	0	0
11	0	1335	0	0
12	0	1335	0	0
13	0	1335	0	0
14	0	1335	0	0
15	0	1335	0	0

After Histogram Equalisation



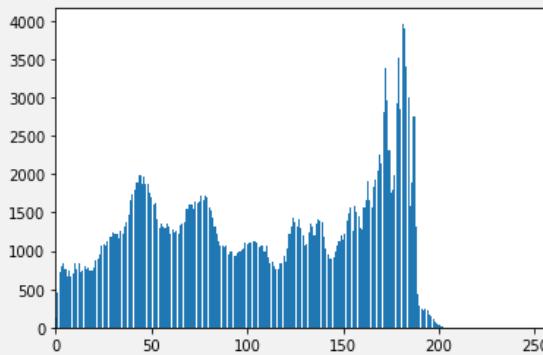
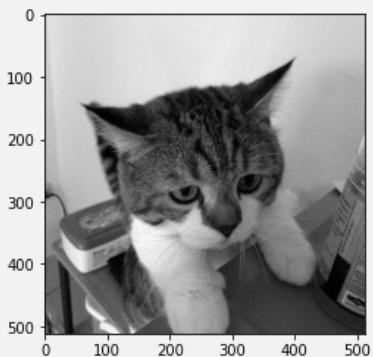
minimize equalize និងបន្តិចតាមលក្ខណៈ contrast ធំជំនួយ

# PYTHON: HISTOGRAM CALCULATION

## I. cv2.calcHist()

```
import cv2
img = cv2.imread("kitty.jpg")
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
hist = cv2.calcHist([img_gray], [0], None, [256], [0, 256])
plt.imshow(img_gray, cmap='gray'); plt.show()
plt.bar(range(0,256), hist[:,0])
plt.xlim([0, 256])
plt.show()
```

channel [0] : grayscale  
mask  
histSize  
range 0 - 255  
channel in 0



### ◆ calcHist() [1/3]

```
void cv::calcHist ( const Mat * images,
                   int nimages,
                   const int * channels,
                   InputArray mask,
                   OutputArray hist,
                   int dims,
                   const int * histSize,
                   const float ** ranges,
                   bool uniform = true,
                   bool accumulate = false
)
```

Python:

```
hist = cv.calcHist( images, channels, mask, histSize, ranges[ hist[, accumulate]] )
```

```
#include <opencv2/imgproc.hpp>
```

Calculates a histogram of a set of arrays.

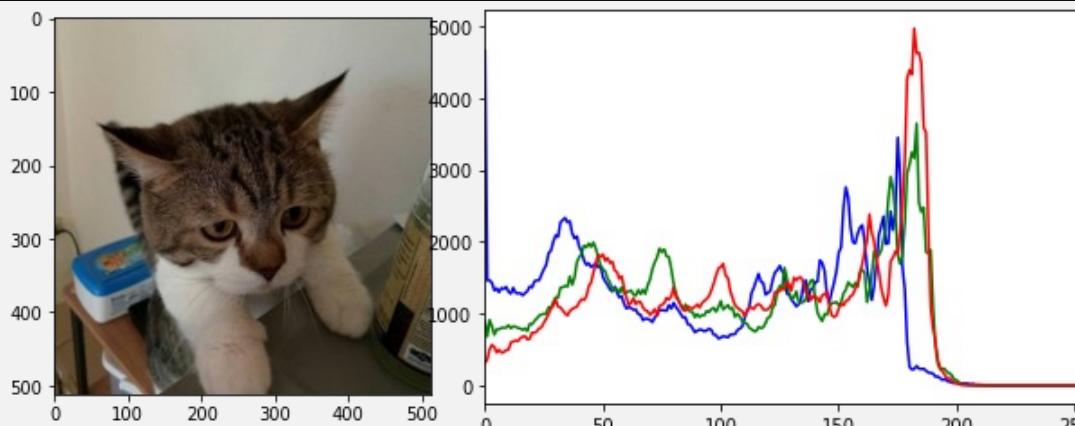
[https://docs.opencv.org/4.5.1/d6/dc7/group\\_\\_imgproc\\_\\_hist.html#ga4b2b5fd75503ff9e6844cc4dcdaed35d](https://docs.opencv.org/4.5.1/d6/dc7/group__imgproc__hist.html#ga4b2b5fd75503ff9e6844cc4dcdaed35d)

# PYTHON: HISTOGRAM CALCULATION

## I. cv2.calcHist() RGB

```
plt.imshow(img[:, :, ::-1]); plt.show()

chans = cv2.split(img)
colors = ("b", "g", "r")
for (chan, color) in zip(chans, colors):
    # create a histogram for the current channel and plot it
    hist = cv2.calcHist([chan], [0], None, [256], [0, 256])
    plt.plot(hist, color=color)
    plt.xlim([0, 256])
plt.show()
```



### ◆ calcHist() [1/3]

```
void cv::calcHist ( const Mat * images,
                   int nimages,
                   const int * channels,
                   InputArray mask,
                   OutputArray hist,
                   int dims,
                   const int * histSize,
                   const float ** ranges,
                   bool uniform = true ,
                   bool accumulate = false
)
```

#### Python:

```
hist = cv.calcHist( images, channels, mask, histSize, ranges[ hist[, accumulate] ] )
```

```
#include <opencv2/imgproc.hpp>
```

Calculates a histogram of a set of arrays.

[https://docs.opencv.org/4.5.1/d6/dc7/group\\_\\_imgproc\\_\\_hist.html#ga4b2b5fd75503ff9e6844cc4dcdaed35d](https://docs.opencv.org/4.5.1/d6/dc7/group__imgproc__hist.html#ga4b2b5fd75503ff9e6844cc4dcdaed35d)

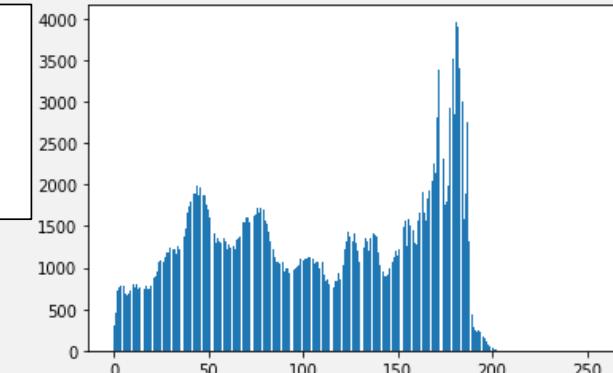
# PYTHON: HISTOGRAM CALCULATION

## 2. np.histogram()

```
img = cv2.imread("kitty.jpg", 0)
hist,bins = np.histogram(img.flatten(),256,[0,256])
plt.bar(range(0,256),hist)
plt.show()
```

### Examples

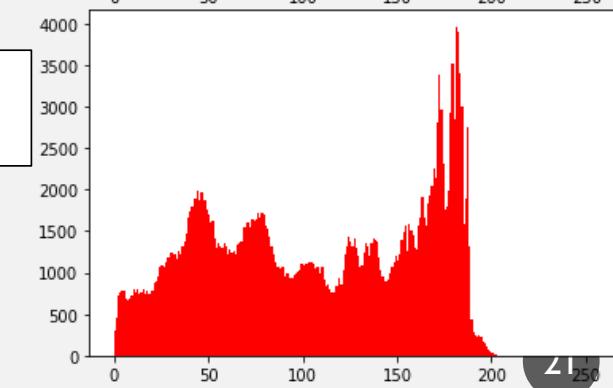
```
>>> a = np.array([[1,2], [3,4]])
>>> a.flatten()
array([1, 2, 3, 4])
```



## 3. plt.hist()

```
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.show()
```

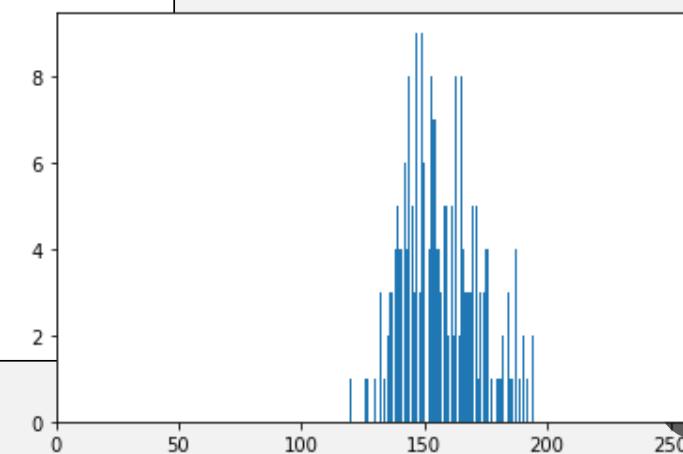
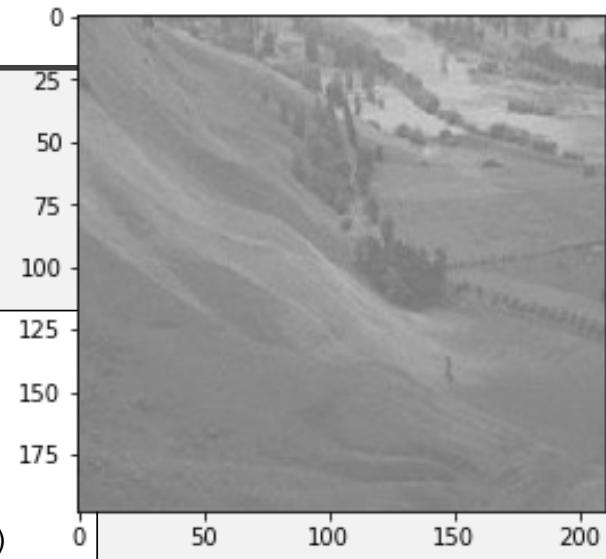
- What do you think about the histogram?



# HISTOGRAM EQUALIZATION

- Histogram Equalization

```
import cv2
img = cv2.imread("area.jpg")
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
hist = cv2.calcHist(img_gray, [0], None, [256], [0, 256])
plt.imshow(img_gray, cmap='gray', vmin=0, vmax=255); plt.show()
plt.bar(range(0, 256), hist[:, 0])
plt.xlim([0, 256])
plt.show()
```



What do you think about the histogram?

# HISTOGRAM EQUALIZATION

## ◆ equalizeHist()

```
void cv::equalizeHist ( InputArray src,  
                      OutputArray dst  
)
```

### Python:

```
dst = cv.equalizeHist( src[, dst] )
```

```
#include <opencv2/imgproc.hpp>
```

Equalizes the histogram of a grayscale image.

The function equalizes the histogram of the input image using the following algorithm:

- Calculate the histogram  $H$  for src .
- Normalize the histogram so that the sum of histogram bins is 255.
- Compute the integral of the histogram:

$$H'_i = \sum_{0 \leq j < i} H(j)$$

- Transform the image using  $H'$  as a look-up table:  $\text{dst}(x, y) = H'(\text{src}(x, y))$

The algorithm normalizes the brightness and increases the contrast of the image.

### Parameters

**src** Source 8-bit single channel image.

**dst** Destination image of the same size and type as src .

### Examples:

[samples/cpp/facedetect.cpp](#).

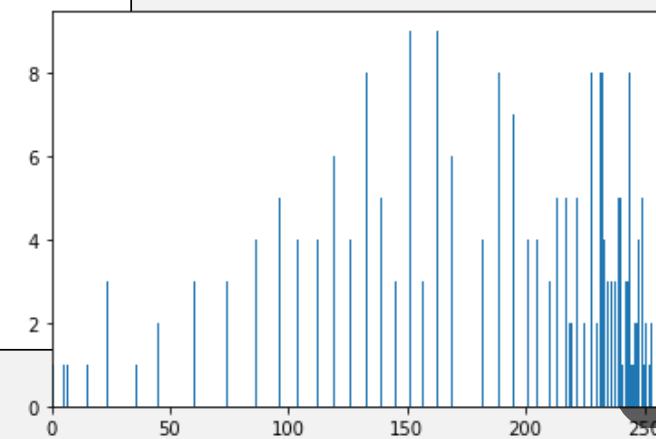
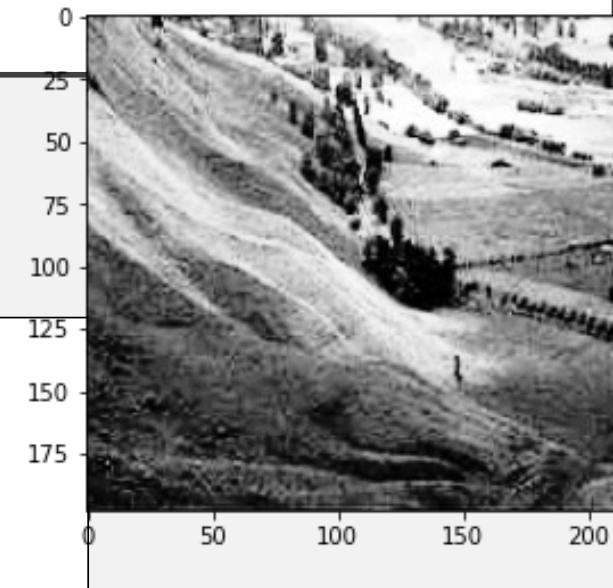
[https://docs.opencv.org/4.5.0/d6/dc7/group\\_\\_imgproc\\_\\_hist.html#ga7e54091f0c937d49bf84152a16f76d6e](https://docs.opencv.org/4.5.0/d6/dc7/group__imgproc__hist.html#ga7e54091f0c937d49bf84152a16f76d6e)

# HISTOGRAM EQUALIZATION

- Histogram Equalization

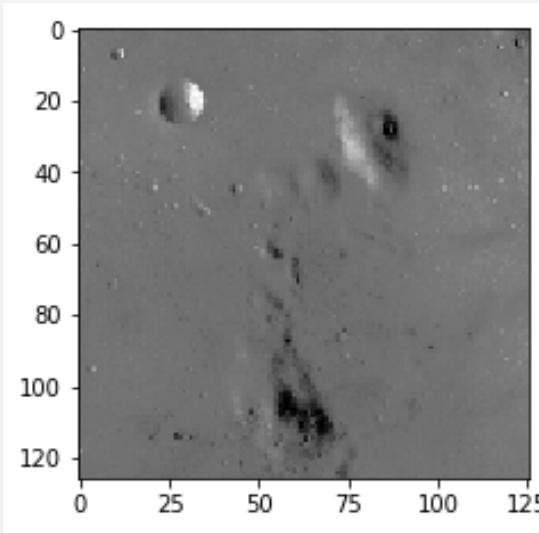
```
import cv2
img = cv2.imread("area.jpg")
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
hist = cv2.calcHist(img_gray, [0], None, [256], [0, 256])
plt.imshow(img_gray, cmap='gray', vmin=0, vmax=255); plt.show()
plt.bar(range(0, 256), hist[:, 0])
plt.xlim([0, 256])
plt.show()

equ = cv2.equalizeHist(img_gray)
plt.imshow(equ, cmap='gray')
plt.show()
```



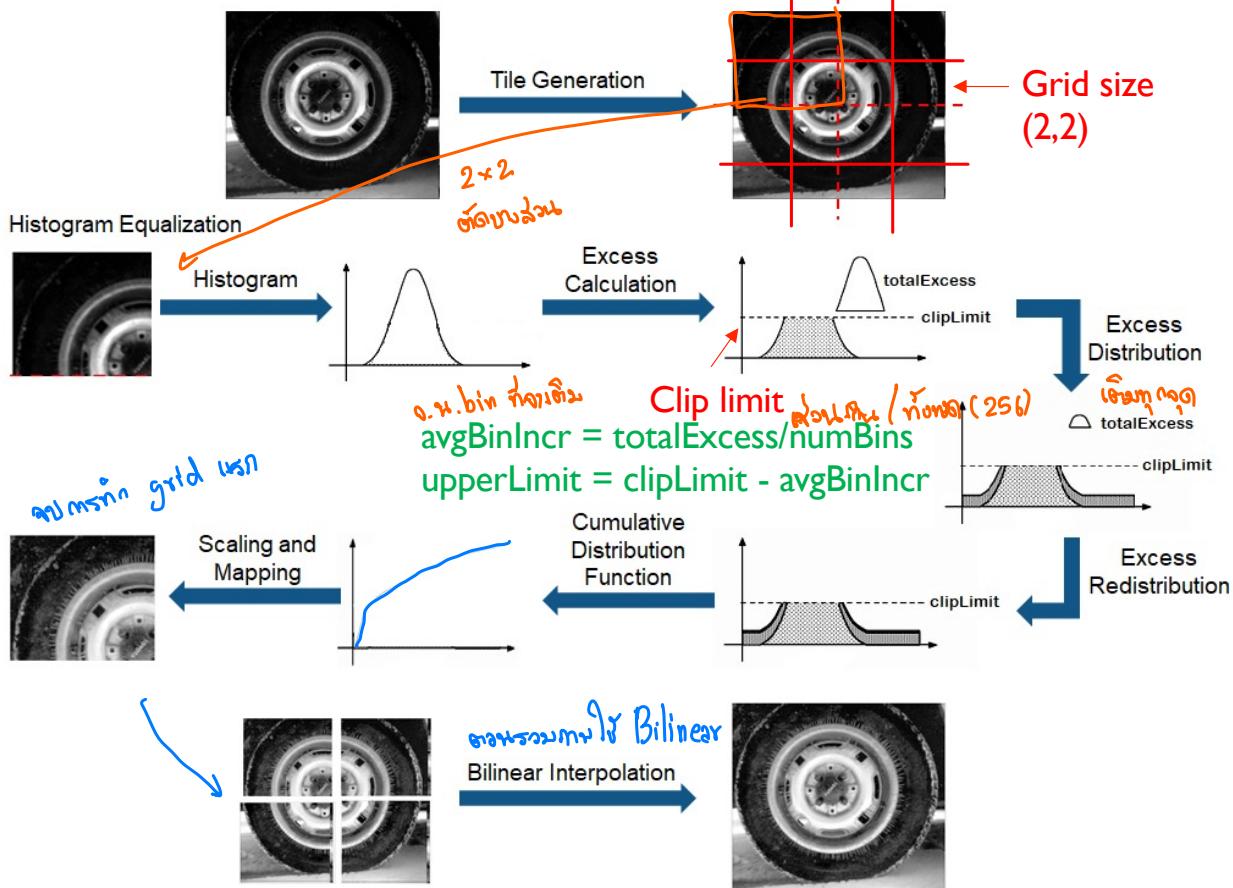
## EXERCISE #1 HISTOGRAM EQU ON SPACE.PNG

- Histogram Equalization
- use **equalizeHist** with 'space.png'. Compare histogram of the original image and the image after histogram equalization. Plot the transformation function from equalization. (only used for 8-bit grayscale!)



# CONTRAST LIMITED ADAPTIVE HISTOGRAM EQUALIZATION (CLAHE)

Karel Zuiderveld, "Contrast Limited Adaptive Histogram Equalization", Graphics Gems IV, p. 474-485, code: p. 479-484.



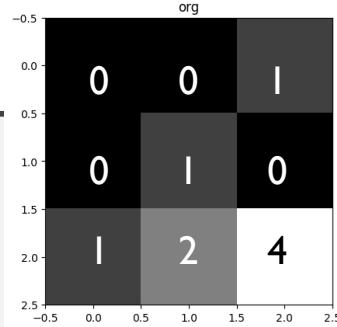
<https://www.mathworks.com/help/visionhdl/ug/contrast-adaptive-histogram-equalization.html#d126e9601>

CLAHE:

1. Image is subdivided into non-overlapping grids. *រួចចំណាំ grid នៅក្នុង ឬ 4x4, 2x2*
2. A histogram is calculated for each grid.
3. Histogram value that *ត្រូវបាន ពាយតាម clip limit ឬ ឬ*
  - is greater than the clip limit, it is replaced with the clip limit.
  - is less than the upper limit, it is increased by the average bin increment *ក្នុងក្នុងក្រុងក្រុង*
  - is between the clip limit and the upper limit, the histogram value is replaced with the clip limit
4. Scale and map new image using cumulative distribution function of the new histogram
5. To combine grids of image together using bilinear interpolation
  - Two parameters *config ឬ ឬ*
    - Clip limit
    - Grid size *ត្រូវបានប្រាកដ ឬ ក្នុង 1x1*

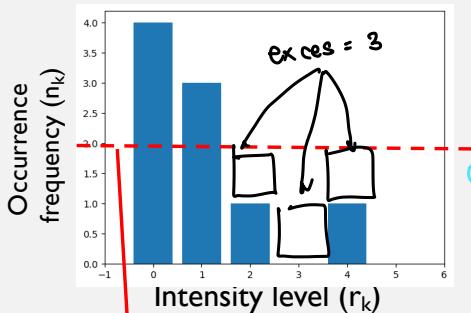
*ទាញយកពី hist equalize*

# CONTRAST LIMITED ADAPTIVE HISTOGRAM EQUALIZATION (CLAHE)



Original Image

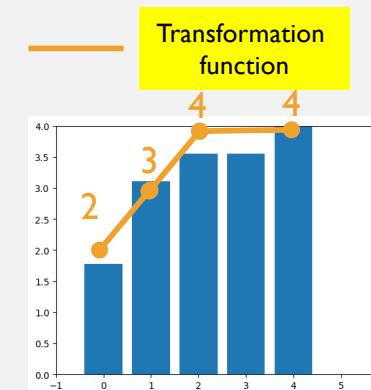
Histogram of the 3x3 image



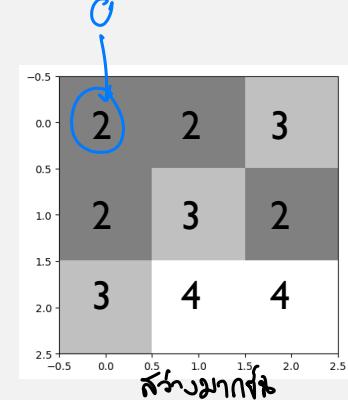
Histogram Equalization

$$\text{avgBinVal} = \frac{\text{excess}}{5 \text{ bin}} = \frac{3}{5} =$$

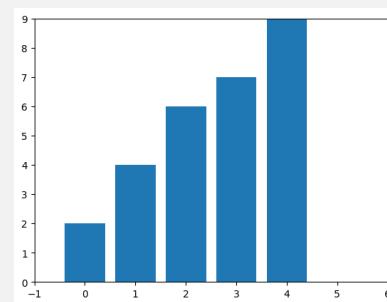
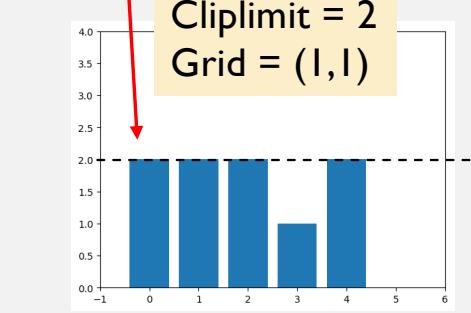
CDF



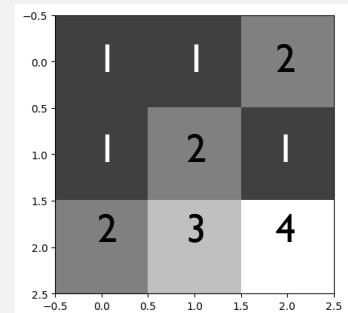
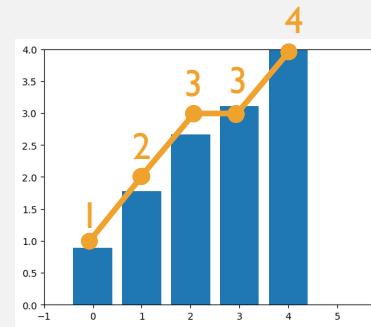
Transformation function



CLAHE



ແລ້ວໃຫຍ່ extreme min hist equalize

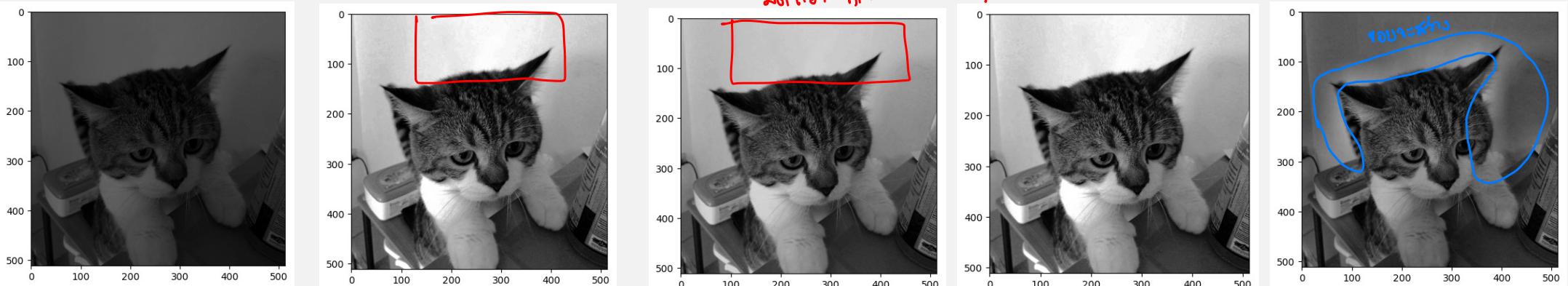


# CONTRAST LIMITED ADAPTIVE HISTOGRAM EQUALIZATION (CLAHE)

$$f_{\text{out}}(x,y) = \frac{f(x,y)}{\max(f(x,y))} + b$$

$f_{\text{out}}(x,y) > 1$   $\rightarrow$   $f_{\text{out}}(x,y) = 1$

Karel Zuiderveld, "Contrast Limited Adaptive Histogram Equalization", Graphics Gems IV, p. 474-485, code: p. 479-484.



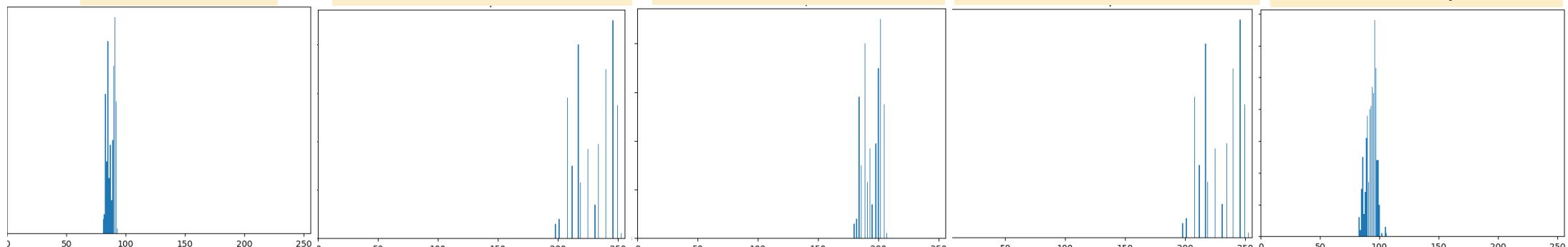
Original Image

Histogram Equalization

CLAHE with cliplimit = 2

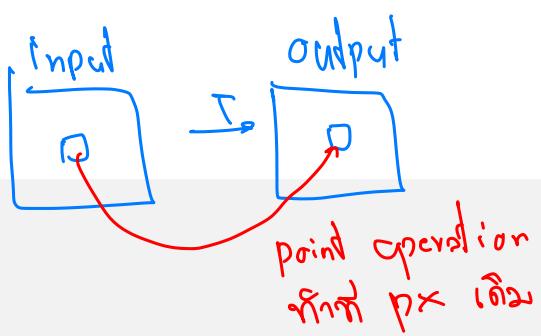
CLAHE with cliplimit = 10

CLAHE with cliplimit = 2



Grid = (1,1) හි grid ග්‍රැසුව

Grid = (8,8)

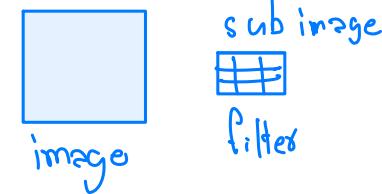


$$S = T(r)$$

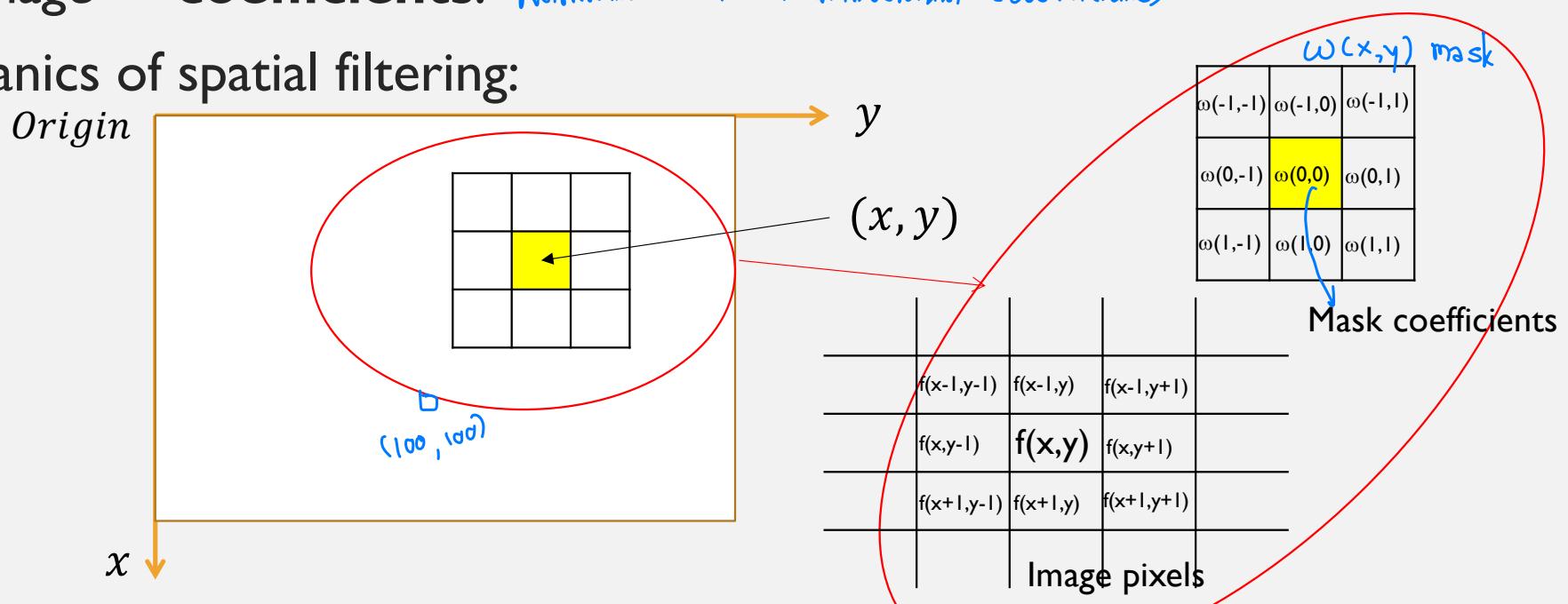
## SPATIAL FILTERING

ກັກກົບ px ສ່າງ ດົວ

# SPATIAL FILTERING

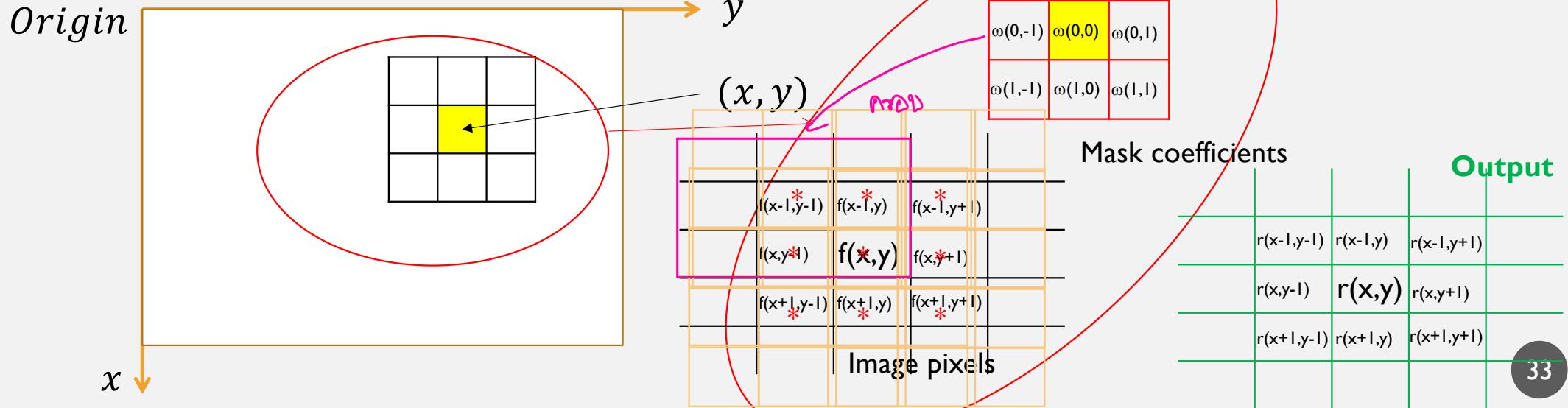


- Neighborhood operations: image pixels and sub-image (neighborhood size); sub-image -> **filter, mask, kernel, template or window**. Values in the sub-image -> **coefficients**. *ເຮັດວຽກທີ່ໃຫຍ່ ຮຶນກຳຕົກຄ່າຂອງຫົ້ນຊັກ (coefficients)*
- Mechanics of spatial filtering:



# SPATIAL FILTERING

- Neighborhood operations: image pixels and sub-image (neighborhood size); sub-image -> **filter, mask, kernel, template or window**. Values in the sub-image -> **coefficients**.
- Mechanics of spatial filtering:



# SPATIAL FILTERING

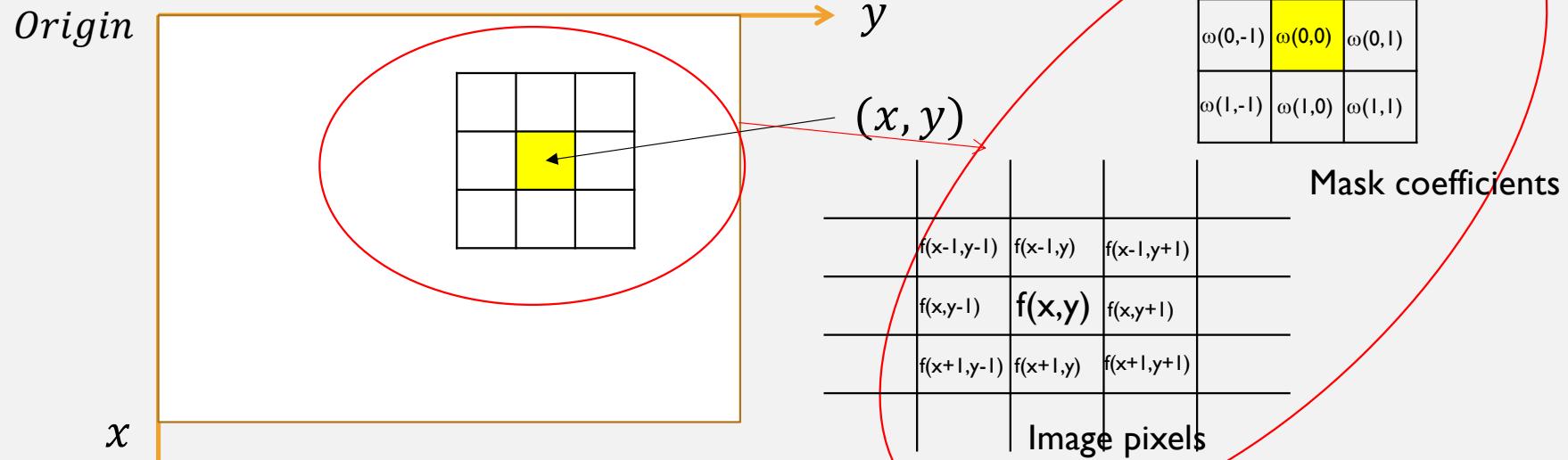
linear filter

- Response of linear filtering - filter mask at a point  $(x,y)$  is

$$R = \underbrace{\omega(-1,-1)f(x-1, y-1) + \omega(-1,0)f(x-1, y) + \dots}_{\text{px units}} + \underbrace{\omega(0,0)f(x, y) + \dots + \omega(1,1)f(x+1, y+1)}_{\text{px units}}$$

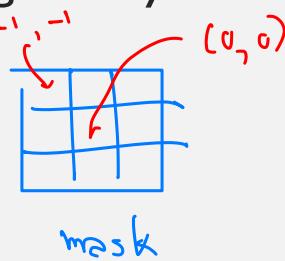
Sum of product von px  
ทั้งหมดของ px

- Mechanics of spatial filtering:



## SPATIAL FILTERING

- In general, linear filtering of an image  $f$  of size  $M \times N$  with a filter mask of size  $m \times n$  is given by the expression:



$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b \omega(s, t) f(x + s, y + t)$$

filter សែរ មុន -1 នូវ 1  
 $m \times n \quad \square - 3 \times 3 = a = \frac{3-1}{2} = 1$   
 $b = 1$

- where  $a = (m-1)/2$ ,  $b = (n-1)/2$
- Linear filtering similar to a frequency domain concept called **convolution**. Spatial filtering referred to as **convolving a mask with an image**. Filter masks can be called **convolution masks (kernel)**. សម្រាប់គេង CNN

# SPATIAL FILTERING

- Simplified notation:

$$R = \sum_{i=1}^{mn} \omega_i z_i$$

$\omega$  are mask coefficients

$z$ : values of image gray levels corresponding those coefficients

$mn$ : the total number of coefficients in the mask.

- **Image border:** ក្រប់ទាញខំ ក្នុងមន្ត្រី ឬ ឯុទ្ធសាស្ត្រ

- partial filter mask នៃ 4 តាមក្រុងមន្ត្រី 4 ពីរក្នុងក្រុងមន្ត្រី

- use 'padding' the image by adding rows and columns of 0's

- padding by replicating rows or columns. ពិនិត្យលាក់លាក់បង្ហាញ

- [perfectly filtered] limit excursions of the centre of the filter mask to distance to less than  $(n-1)/2$  pixels from the border.



1	1	2	3
1	2	3	
1	1	2	

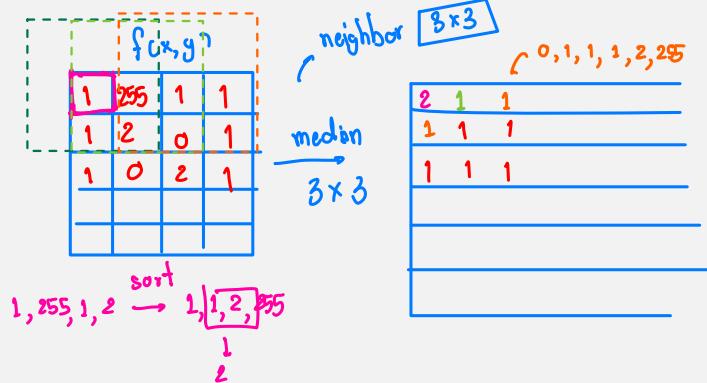
X	X	X	
X	0	0	
X	0	0	

មួយទីតាំង

# SPATIAL FILTERING

- Non-linear filters:
  - Operate on neighborhoods
  - Filtering operations **conditionally** based on the value of neighborhood pixels under consideration. (not just use sum-of-products manner) e.g.,

## median filters



ກຳນົດຂອງກ່າວເປັນ noise ຊົ່ວ ?

0, 255  
↓      ↓  
pepper   salt  
noise   noise

## SPATIAL FILTERING

- 1) Smoothing Linear Filters
- 2) Sharpening Spatial Filters ຄມ່ງຈຳນວ

## SMOOTHING SPATIAL FILTERS

- Used for blurring and for noise reduction.
- Blurring – preprocessing steps, such as removal of small details from an image prior to object extraction, and bridging of small gaps in lines or curves.
- I) **Smoothing linear filters**
- 2) **Order statistical filters**

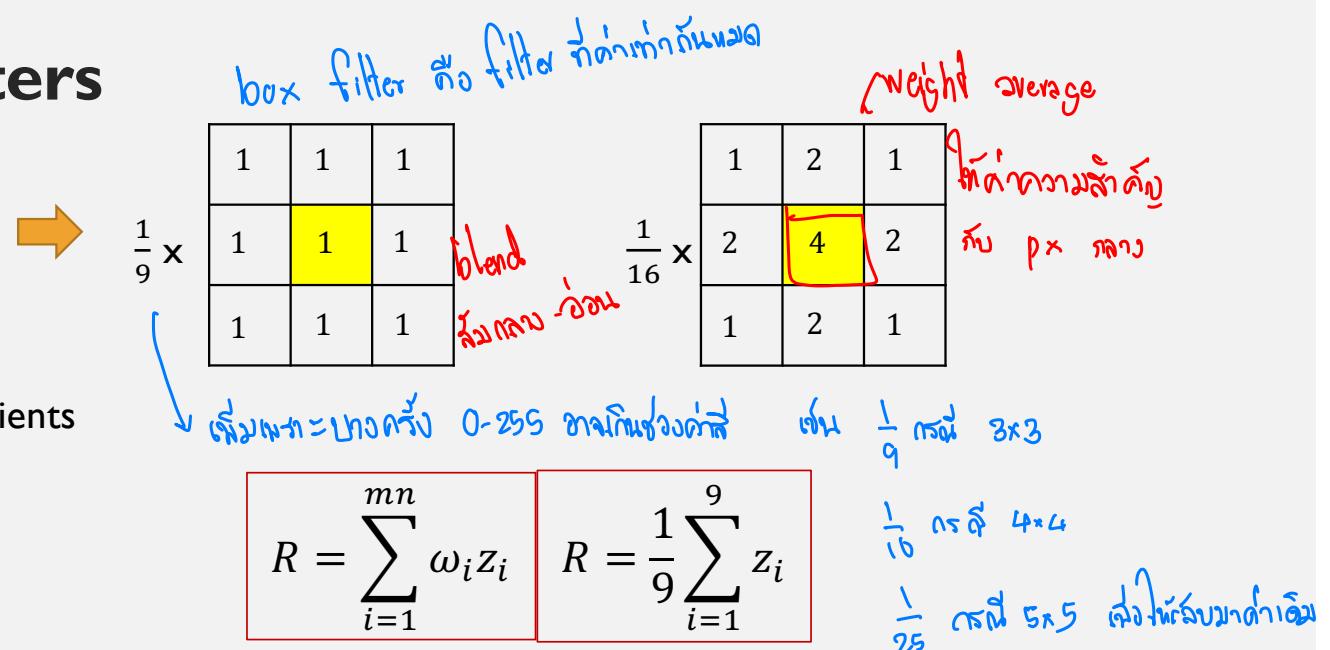
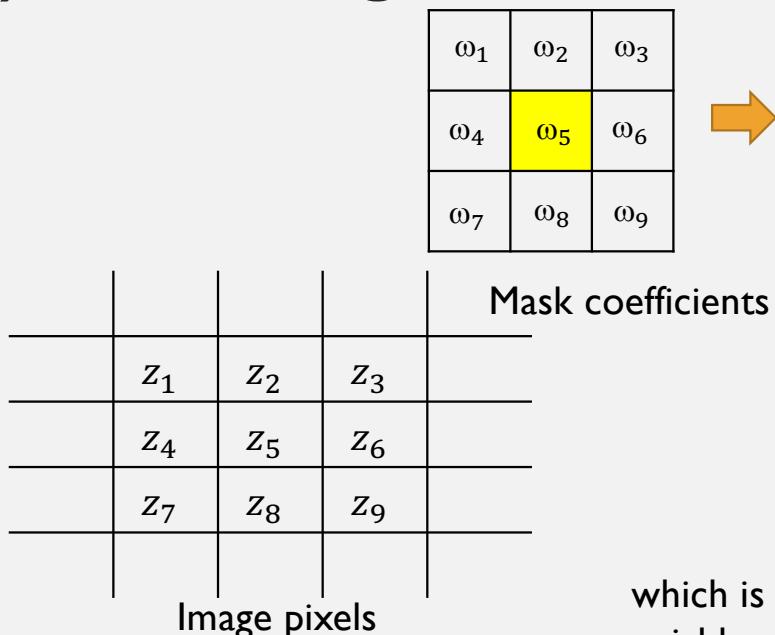
សំគាល់ការណែនក្នុង

## **SMOOTHING SPATIAL FILTERS**

- **I) Smoothing linear filters**
  - Averaging filters or lowpass filters.
  - Method: replacing the value of every pixel in an image by the average of the gray levels in the neighborhood defined by the filter mask.
  - Reduce ‘sharp’ transitions in gray levels.
  - Noise reduction - Noise typically has sharp transitions in gray levels.
  - Undesirable side effect – blur edges.

# SMOOTHING SPATIAL FILTERS

- I) Smoothing linear filters



which is the average of the gray levels of the pixels in the  $3 \times 3$  neighborhood

-A spatial averaging filter -> **box filter**

-the right mask ( $1/16$ ) called **weighted average**

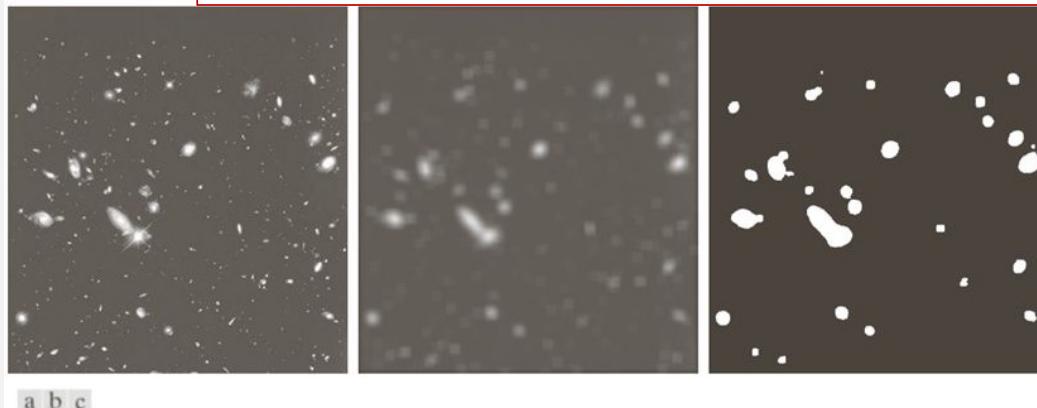
## SMOOTHING SPATIAL FILTERS

- I) Smoothing linear filters
- General implementation  $M \times N$  image with a weight averaging filter of size  $m \times n$  is given by the expression:

$$g(x, y) = \frac{\sum_{s=-a}^a \sum_{t=-b}^b \omega(s, t) f(x + s, y + t)}{\sum_{s=-a}^a \sum_{t=-b}^b \omega(s, t)}$$

original

q, 1b, ...



**FIGURE 3.34** (a) Image of size  $528 \times 485$  pixels from the Hubble Space Telescope. (b) Image filtered with a  $15 \times 15$  averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

© Gonzalez & Woods, Digital Image Processing



**Fig. 3.35** (a) Original image, of size  $500 \times 500$  pixels. (b) – (f) Result of smoothing with square averaging filter masks of size  $n = 3, 5, 9, 15$ , and 35 respectively.

## SMOOTHING SPATIAL FILTERS

- **2) Order statistical filters**

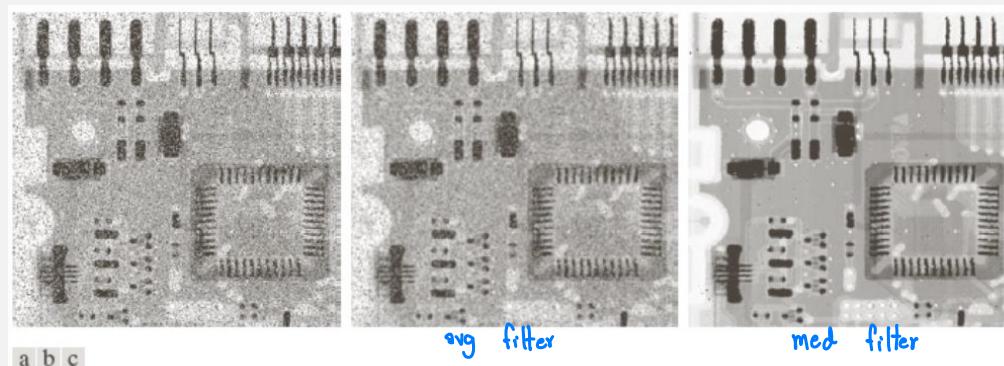
- non-linear filters – involved ranking process, e.g., median filter (replace the centre with the median of the gray levels in the neighborhood of that pixel.)  
*median filter*
- Principal function is to force points with distinct gray levels to be more like their neighbors.
- Effective in presence of impulse noise (salt and pepper noise).  
*Gaussian noise ค่าสูงหรือต่ำมากๆ*

*Gaussian noise សង្ឃ័េជ្ជ noise រួមអ៊ីតុកនឹង*

## SMOOTHING SPATIAL FILTERS

- **2) Order statistical filters**

- X-ray image of a circuit board corrupted by impulse noise.
- Averaging filters: less visible noise but blurring
- Median filters: better suited for removal the impulse noise.



**FIGURE 3.35** (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a  $3 \times 3$  averaging mask. (c) Noise reduction with a  $3 \times 3$  median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

© Gonzalez & Woods, Digital Image Processing

## **SPATIAL FILTERING**

- 1) Smoothing Linear Filters
- 2) Sharpening Spatial Filters

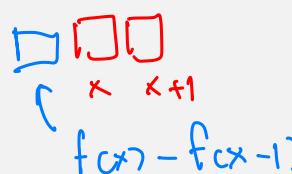


## SHARPENING SPATIAL FILTERS

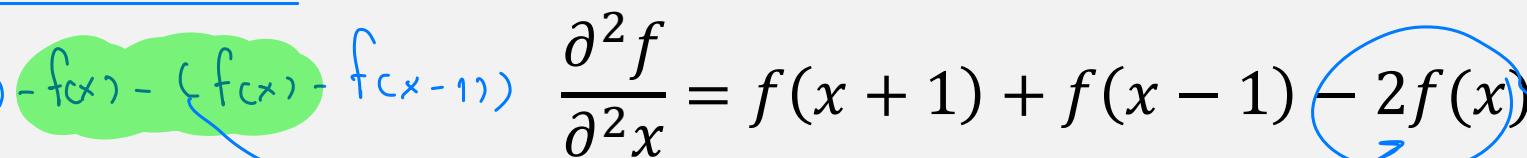
- To highlight fine detail in an image or to enhance detail that has been blurred.
- Used in electronic printing and medical imaging, industrial inspection and autonomous guidance in military systems.
- Accomplished by spatial differentiation; the strength of the response of a derivative operator  $\approx$  degree of discontinuity of the image.
- Enhance edge and other discontinuities (such as noise) and deemphasizes areas with slowly varying gray-level values.

## SHARPENING SPATIAL FILTERS

- Derivatives of a digital function are defined in terms of differences.
- Digital values:** finite and the shortest distance over which that change can occur is between adjacent pixels.

- Definition of the first-order derivative of a 1-D  $f(x)$  is  $\frac{\partial f}{\partial x} = f(x + 1) - f(x)$   
  
ท้าทายสื่อสารด้วยรูปสัญญาณ

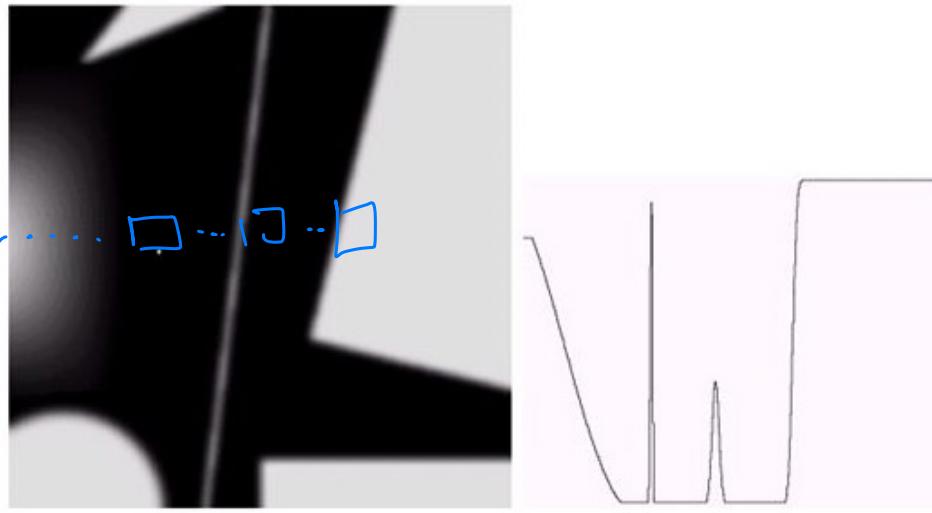
- Second-order derivative as the difference: คุณสมบัติของฟังก์ชัน

$$\frac{\partial^2 f}{\partial^2 x} = f(x + 1) + f(x - 1) - 2f(x)$$


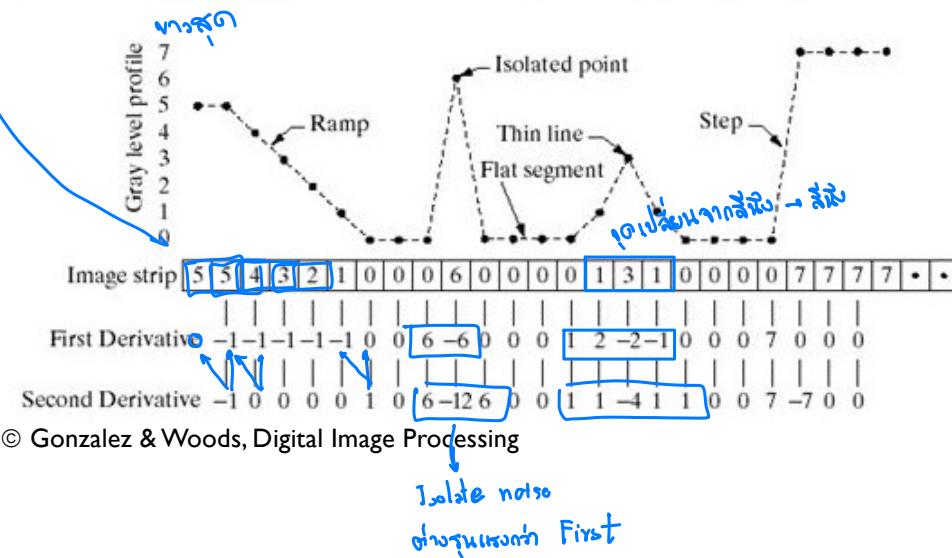
# SHARPENING SPATIAL FILTERS

a  
b  
c

**FIGURE 3.38**  
 (a) A simple image. (b) 1-D horizontal gray-level profile along the center of the image and including the isolated noise point. (c) Simplified profile (the points are joined by dashed lines to simplify interpretation).



0 - 7



- **Ramps:** first-order is nonzero all along, second-order only at the onset and end of the ramp.
- **Isolated noise:** second-order much more aggressive in enhancing sharp changes.
- **Thin line:** fine details (second-order stronger)
- First-order – thicker edges,
- Second-order – stronger response to fine detail & produce double response at step changes.

# SHARPENING SPATIAL FILTERS

- I) Second-derivatives (Laplacian)

- Isotropic filters – independent of the direction of the discontinuities in the image + rotation invariant.
- Laplacian of  $f(x,y)$ :

$$\frac{\partial^2 f}{\partial^2 x_{\text{row}}} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial^2 y_{\text{column}}} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

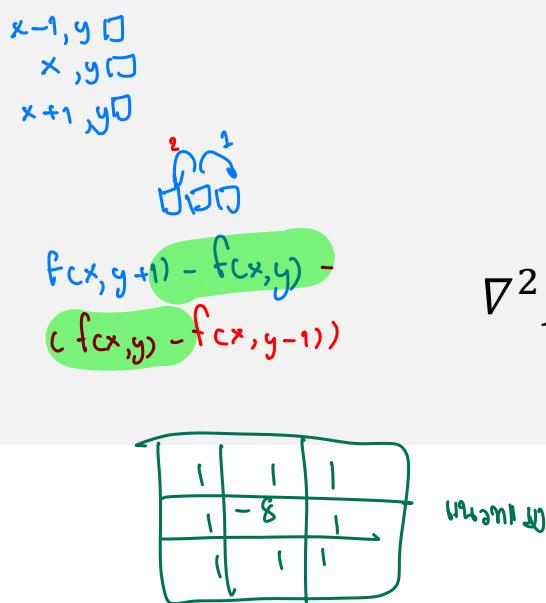
$$\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y)$$

กลับฟื้นฟู

0	-1	0
-1	4	-1
0	-1	0

Laplacian  
filter

0	1	0
1	-4	1
0	1	0



## SHARPENING SPATIAL FILTERS

- I) Second-derivatives (Laplacian) (cont')

0	I	0
I	-4	I
0	I	0

I	I	I
I	-8	I
I	I	I

0	-I	0
-I	4	-I
0	-I	0

-I	-I	-I
-I	8	-I
-I	-I	-I

- Highlights gray-level discontinuities in an image and deemphasizes regions (slowly varying gray levels).
- Combining original and Laplacian images – sharpening effect and with background features

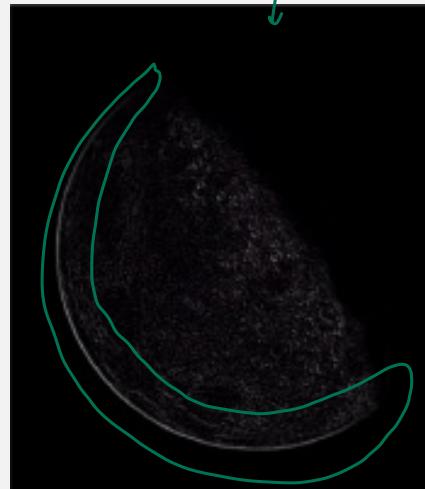
# SHARPENING SPATIAL FILTERS

- I) Second-derivatives (Laplacian) (cont')



Original image

+



Laplacian image



Combined image

- Combining original image and the Laplacian image:
  - the detail in this image is clearer and sharper than in the original image.
  - increasing the contrast at the locations of gray-level discontinuities.

## SHARPENING SPATIAL FILTERS

- The basic way in which we use the Laplacian for image enhancement is as follows:

$$g(x, y) = \begin{cases} f(x, y) - \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is negative} \\ f(x, y) + \nabla^2 f(x, y) & \text{if the center coefficient of the Laplacian mask is positive} \end{cases}$$

0	-1	0
-1	4+1	-1
0	-1	0

for example,

$$\begin{aligned} g(x, y) &= f(x, y) - [f(x+1, y) + f(x-1, y) \\ &\quad + f(x, y+1) + f(x, y-1) - 4f(x, y)] \\ &= 5f(x, y) - [f(x+1, y) + f(x-1, y) \\ &\quad + f(x, y+1) + f(x, y-1)] \end{aligned}$$

## SHARPENING SPATIAL FILTERS

- 2) First-derivatives ສົງຄາງຄືຂັ້ນໜີການ
- The gradient where the gradient of function  $f(x,y)$  is defined as 2D column vector:

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

horizontal edge  
vertical edge

- The magnitude:

$$\begin{aligned}\nabla f = mag(\nabla f) &= [G_x^2 + G_y^2]^{1/2} \\ &= \left[ \left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2 \right]^{1/2}\end{aligned}$$

(ນອບໃມ່ x) + (ນອບໃມ່ y)

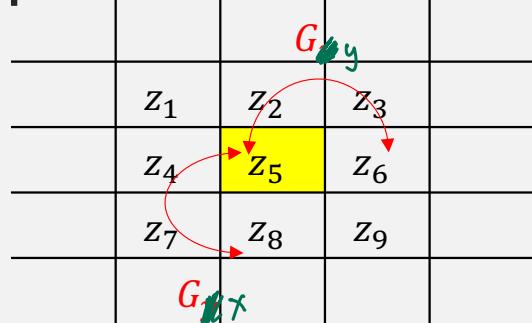
$$\nabla f \approx |G_x| + |G_y|$$

## SHARPENING SPATIAL FILTERS

- 2) First-derivatives
- To reduce computational burden, use absolute values instead:

$$\nabla f \approx |G_x| + |G_y|$$

- Simpler and reserves relative changes in gray levels.



$$G_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} z_5 & z_6 \\ z_8 & z_9 \end{bmatrix}$$

$$\begin{array}{|c|c|} \hline z_5 & z_6 \\ \hline -1 & 1 \\ \hline z_8 & z_9 \\ \hline \end{array}$$

Early development, Roberts [1965]:

$$G_x = (z_9 - z_5)$$

$$G_y = (z_8 - z_6)$$

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|$$

Robert cross-gradient operators:

# SHARPENING SPATIAL FILTERS

$G_x$

- Robert cross-gradient operators:

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|$$

0	-1
1	0

-1	0
0	1

- Even size – awkward to implement អារាល់ត្រូវមានចាត់បន្ថែមទាំងពីរ
- Sobel operators:

$$\nabla f \approx |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

$z_1$	$z_2$	$z_3$
$z_4$	$z_5$	$z_6$
$z_7$	$z_8$	$z_9$

$G_x$

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

ខ្សែករុកស្រាវលើទៅក្នុង 2 ពេល

ខាងក្រោមនៃ ឈុយនន៍ -&

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

$G_y$

$G_x$  detects horizontal edges;  
 $G_y$  for vertical edges.  
Sometimes,  $G_x$  and  $G_y$  concepts are swap. (OpenCV library)

## Sobel filter

$$\begin{array}{|c|c|c|c|} \hline & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

$$G_x \text{ 卷积} \quad \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 3 & 4 & 4 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

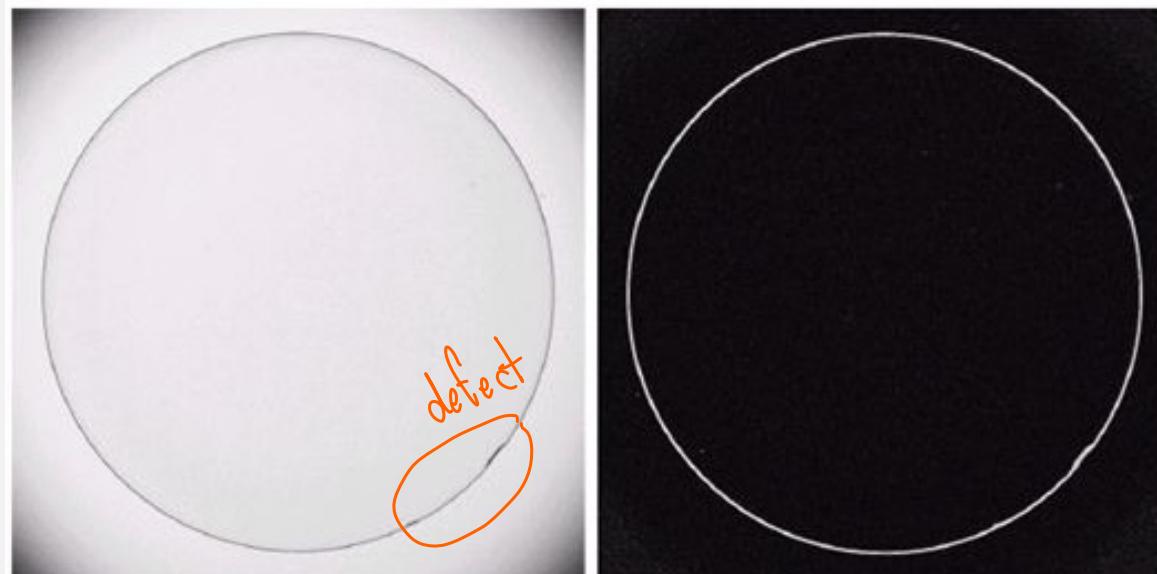
$$\begin{array}{|c|c|c|c|} \hline & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

$$G_y \text{ 卷积} \quad \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

$$\text{卷积} \quad \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 1 & 0 & 0 \\ \hline 2 & 0 & 0 \\ \hline \end{array}$$

## SHARPENING SPATIAL FILTERS

- **Applications:**
- Aid human in the detection of defects: preprocessing step.



a b

**FIGURE 3.45**  
Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock).  
(b) Sobel gradient.  
(Original image courtesy of Mr. Pete Sites, Perceptics Corporation.)

# COMBINING SPATIAL ENHANCEMENT METHODS

- Whole body bone scan for detecting bone infection or tumors.



(a)  
Original image

+



(b)  
Laplacian of (a)  $\nabla^2$

=

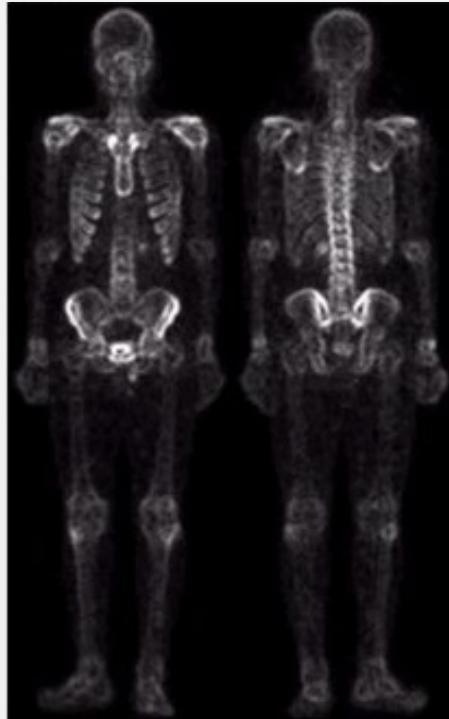


(c)  
Sharpening  
by (a) + (b)



(d)  
Sobel of (a)  
 $\nabla (a)$

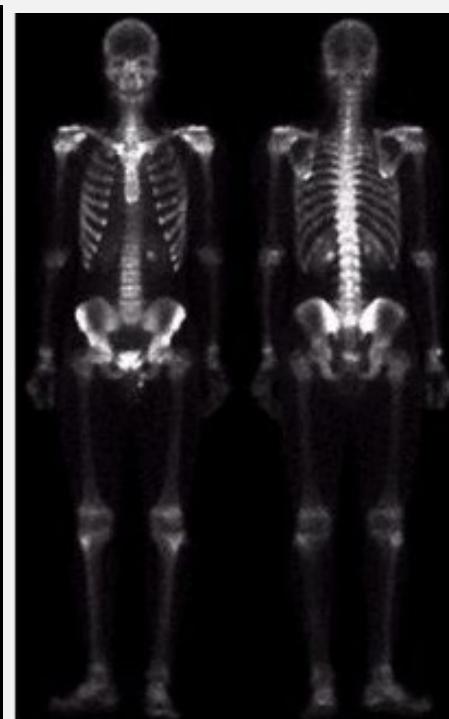
# COMBINING SPATIAL ENHANCEMENT METHODS



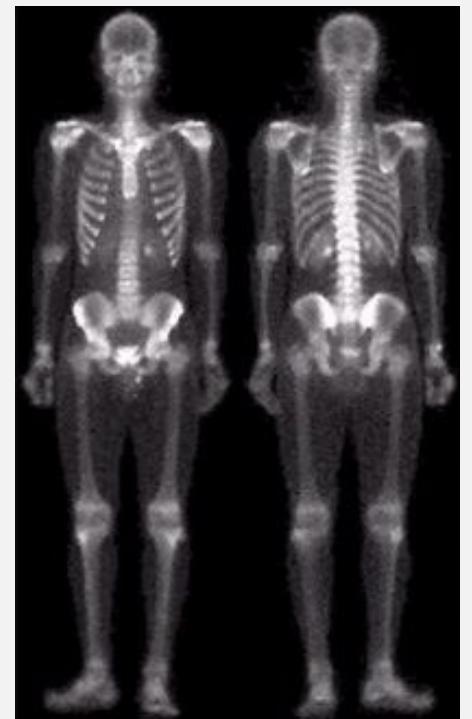
(e)  
(d) Smoothed by  
an average filter



(f)  
Mask from product of  
(c) and (e)



(g)  
sum of (a) and (f)



(h)  
Power-law transformation  
on (g)

## PYTHON: SPATIAL FILTERING

- e is gray value of the current pixel in the original image.

$$\begin{array}{|c|c|c|} \hline & a & b & c \\ \hline d & e & f & \\ \hline g & h & i & \\ \hline \end{array} \quad \frac{1}{9} \times \begin{array}{|c|c|c|} \hline | & | & | \\ \hline | & | & | \\ \hline | & | & | \\ \hline \end{array}$$

- The average is the gray value of the corresponding pixel in the new image

$$\frac{1}{9} (a + b + c + d + e + g + h + i)$$

## SPATIAL FILTERING

- 1) create a filter (kernel)

```
kernel = np.ones((3, 3), np.float32) / 9
```

- 2) apply on the image
- Using **filter2D** (2-D digital filter)

```
dst = cv2.filter2D(img, -1, kernel)
```

# SPATIAL FILTERING

## • filter2D

Python:

```
cv.filter2D( src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]] ) -> dst
```

```
#include <opencv2/imgproc.hpp>
```

Convolves an image with the kernel.

The function applies an arbitrary linear filter to an image. In-place operation is supported. When the aperture is partially outside the image, the function interpolates outlier pixel values according to the specified border mode.

The function does actually compute correlation, not the convolution:

$$dst(x, y) = \sum_{\substack{0 \leq x' < \text{kernel.cols} \\ 0 \leq y' < \text{kernel.rows}}} \text{kernel}(x', y') * \text{src}(x + x' - \text{anchor.x}, y + y' - \text{anchor.y})$$

That is, the kernel is not mirrored around the anchor point. If you need a real convolution, flip the kernel using [flip](#) and set the new anchor to `(kernel.cols - anchor.x - 1, kernel.rows - anchor.y - 1)`.

The function uses the DFT-based algorithm in case of sufficiently large kernels (~ 11 x 11 or larger) and the direct algorithm for small kernels.

### Parameters

<b>src</b>	input image.
<b>dst</b>	output image of the same size and the same number of channels as src.
<b>ddepth</b>	desired depth of the destination image, see <a href="#">combinations</a>
<b>kernel</b>	convolution kernel (or rather a correlation kernel), a single-channel floating point matrix; if you want to apply different kernels to different channels, split the image into separate color planes using <code>split</code> and process them individually.
<b>anchor</b>	anchor of the kernel that indicates the relative position of a filtered point within the kernel; the anchor should lie within the kernel; default value (-1,-1) means that the anchor is at the kernel center.
<b>delta</b>	optional value added to the filtered pixels before storing them in dst.
<b>borderType</b>	pixel extrapolation method, see <a href="#">BorderTypes</a> . <b>BORDER_WRAP</b> is not supported.

### Enumerator

**BORDER\_CONSTANT**

Python: cv.BORDER\_CONSTANT

iiiiii|abcdefg|iiiiii with some specified i

**BORDER\_REPLICATE**

Python: cv.BORDER\_REPLICATE

aaaaaa|abcdefg|hhhhhh

**BORDER\_REFLECT**

Python: cv.BORDER\_REFLECT

fedcba|abcdefg|hgfedcb

**BORDER\_WRAP**

Python: cv.BORDER\_WRAP

cdefgh|abcdefg|abcdefg

**BORDER\_REFLECT\_101**

Python: cv.BORDER\_REFLECT\_101

gfedcb|abcdefg|gfedcba

**BORDER\_TRANSPARENT**

Python: cv.BORDER\_TRANSPARENT

uvwxyz|abcdefg|ijklmno

**BORDER\_REFLECT101**

Python: cv.BORDER\_REFLECT101

same as BORDER\_REFLECT\_101

**BORDER\_DEFAULT**

Python: cv.BORDER\_DEFAULT

same as BORDER\_REFLECT\_101

**BORDER\_ISOLATED**

Python: cv.BORDER\_ISOLATED

do not look outside of ROI

## SPATIAL FILTERING

```
small_im = np.random.random((5,5))*100
small_im = small_im.astype(np.uint8)

small_im
array([[71, 28, 50, 47, 59],
       [60, 68, 8, 75, 3],
       [50, 23, 87, 41, 77],
       [67, 32, 56, 50, 2],
       [57, 45, 4, 89, 69]], dtype=uint8)

kernel = np.ones((3,3),np.float32)/9
array([[ 0.11111111,  0.11111111,  0.11111111],
       [ 0.11111111,  0.11111111,  0.11111111],
       [ 0.11111111,  0.11111111,  0.11111111]], dtype=float32)

dst_im = cv2.filter2D(small_im,-1,kernel)
array([[58, 47, 47, 36, 51],
       [47, 49, 47, 50, 52],
       [47, 50, 49, 44, 46],
       [42, 47, 47, 53, 56],
       [45, 46, 46, 42, 50]], dtype=uint8)
```

- **magic(n)** returns an n-by-n matrix constructed from the integers 1 through  $n^2$  with equal row and column sums

## AVERAGING AN IMAGE

```
img = cv2.imread("camaraman.tif")  
  
kernel = np.ones((3,3),np.float32)/9  
dst = cv2.filter2D(img,-1,kernel)
```

```
plt.subplot(121),plt.imshow(img),plt.title('Original')  
plt.xticks([]), plt.yticks([])  
plt.subplot(122),plt.imshow(dst),plt.title('Averaging')  
plt.xticks([]), plt.yticks([])  
plt.show()
```



-What do you see?  
-what if you change filter size?

# SOBEL FILTER

```
img = cv2.imread("sudoku.jpg", 0)
sobelx = cv2.Sobel(img, cv2.CV_8U, 1, 0, ksize=3)
sobely = cv2.Sobel(img, cv2.CV_8U, 0, 1, ksize=3)
```



Gy      ↓ ក្រែងអនុគមន៍ខាងក្រោម

Gx      →

## **EXERCISE #2 SPATIAL FILTERING**

- Create a sharpening filter and apply on ‘cameraman.tif’ image.
- Hint: see  $3 \times 3$  Laplacian filter

## **EXERCISE #3 CAN YOU ENHANCE THIS IMAGE?**



- moon.png

## REFERENCES

- Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Addison-Wesley
- Chapter 3, Section 3.3-3.8